Online Negotiation for Privacy Preserving Data Publishing

Alexandra Pilalidou

MASTER THESIS

—♦—

Ioannina, July 2010

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF IOANNINA

ONLINE NEGOTIATION FOR PRIVACY PRESERVING DATA PUBLISHING

Η
ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

Υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης
του Τμήματος Πληροφορικής
Εξεταστική Επιτροπή

από την

Αλεξάνδρα Πιλαλίδου

ως μέρος των Υποχρεώσεων

για τη λήψη

του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΟ ΛΟΓΙΣΜΙΚΟ

Ιουλιος 2010

DEDICATION

This thesis is dedicated to my family for supporting me all the way since the beginning of my studies.

ACKNOWLEDGMENTS

I am thankful to my supervisor Dr. Panos Vassiliadis for guiding, encouraging and motivating me throughout this research work.

I also would like to thank all my friends and colleagues for their help and encouragement throughout this work.

Finally, I would like to thank Xiaokui Xiao and Yufeio Tao for the publication of the IPUMS data set as well as for giving us more detailed information for the description of the IPUMS data set.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ΕΚΤΕΝΗΣ ΠΕΡΙΛΗΨΗ ΣΤΑ ΕΛΛΗΝΙΚΑ

Αλεξάνδρα Πιλαλίδου του Αλέξανδρου και της Βαλεντίνης. Msc, Τμήματος Πληροφορικής Πανεπιστήμιο Ιωαννίνων, Ιούλιος 2010. Online negotiation for privacy preserving data publishing.
Επιβλέποντας: Πάνος Βασιλειάδης.

Το πρόβλημα της προστασίας της ιδιωτικής των δεδομένων που δημοσιεύονται ορίζεται ως πρόβλημα της δημόσιας παρουσίασης δεδομένων γύρω από τις δραστηριότητες ή τις πράξεις από ένα σύνολο ατόμων, προκειμένου να εξυπηρετηθούν οι ακόλουθοι δύο ανταγωνιστικοί στόχοι: (α) να επιτρέψουμε σε ένα σύνολο καλοπροαίρετων χρηστών να εφαρμόζουν διάφορους αλγορίθμους εξόρυξης δεδομένων με σκοπό την εξαγωγή χρήσιμων πληροφοριών στατιστικού χαρακτήρα για το σύνολο δεδομένων, και, (β) να εμποδίσουμε έναν κακόβουλο εισβολέα να συνδυάσει κάποιες εξωτερικές πληροφορίες που έχει (στο αίσθηση της προσωπικής γνώσης του εισβολέα, άλλες δημόσια διαθέσιμα σύνολα δεδομένων, κλπ.), προκειμένου να συνδυάσει το συγκεκριμένο πρόσωπο στον πραγματικό κόσμο (και ιδίως των ευαίσθητων πληροφοριών γύρω από αυτό το πρόσωπο) με αντίστοιχη εγγραφή που δημοσιεύεται στο ευρύ κοινό. Η κύρια τεχνική που χρησιμοποιείται για την προστασία αυτών των δεδομένων είναι η ανωνυμοποίηση, η οποία μετατρέπει τα δεδομένα σε μια συγκεκριμένη μορφή πριν δημοσιευθεί. Στην παρούσα εργασία η τεχνική η οποία χρησιμοποιήσαμε για να πετύχουμε την ανωνυμοποίηση ονομάζεται καθολική κωδικοποίηση, η οποία (α) είναι πολύ καλή για την χρήση των αλγόριθμων εξόρυξης δεδομένων που χρησιμοποιεί ο καλοπροαίρετος χρήστης, (β) πολύ γρήγορη σε σχέση με κάποιες άλλες μεθόδους που υπάρχουν στην βιβλιογραφία, συγχρόνως όμως, (γ) υπάρχει το πρόβλημα της διαγραφής κάποιων εγγραφών, με σκοπό να πετύχουμε το επιθυμητό επίπεδο γενίκευσης.

Στην εργασία αυτή αντιμετωπίσαμε τα ακόλουθα προβλήματα που δεν υπήρχαν στην σχετική βιβλιογραφία. Ο πρώτος στόχος ήταν να μελετήσουμε την σύνδεση που έχουν μεταξύ τους οι τρεις παράμετροι του προβλήματος – δηλαδή, το σύνολο των εγγραφών που διαγράφουμε, το επίπεδο γενίκευσης και τέλος το κριτήριο ανωνυμοποίησης. Ο βασικός στόχος της εργασίας είναι να παρέχει στο χρήστη την δυνατότητα της online διαπραγμάτευσης των τριών παραμέτρων που αναφέραμε πάνω, δηλαδή, (α) το επίπεδο της ανωνυμοποίησης που επιθυμεί, (β) το πλήθος των διαγραφόμενων εγγραφών που επιτρέπει, και (γ) το βαθμό της ανωνυμοποίησης που επιθυμεί.

Η πρώτη προσέγγιση που έχουμε είναι ο προϋπολογισμός του ιστογράμματος για όλους τους διάφορους συνδυασμούς ανωνυμοποίησης που μπορεί να κατασκευάσει μια μέθοδος καθολικής κωδικοποίησης. Αυτό επιτρέπει τον υπολογισμό επακριβών λύσεων εξαιρετικά γρήγορα (σε χρόνο μερικών milliseconds). Παρέχουμε στο χρήστη και επακριβείς απαντήσεις (αν μπορούν να υπάρξουν), αλλά και προτάσεις για προσεγγιστικές λύσεις μέσω αυτών των ιστογραμμάτων. Παρ' όλα αυτά, η μέθοδος αυτή προϋποθέτει ένα χρόνο προ-επεξεργασίας για την κατασκευή των ιστογραμμάτων, ο οποίος ανέρχεται στην τάξη μεγέθους μερικών δεκάδων λεπτών – έτσι, υπάρχει χώρος για περαιτέρω βελτιώσεις. Για το σκοπό αυτό προτείνουμε και μία δεύτερη μέθοδο, η οποίο προϋπολογίζει μόνο ένα μικρό ποσοστό των ιστογραμμάτων, με σκοπό να επιταχυνθεί ο χρόνος προ-επεξεργασίας. Τα πειράματά μας έδειξαν γραμμική επιτάχυνση στο χρόνο αυτό, με πολύ καλές ή έστω αποδεκτές τιμές για την ποιότητα του αποτελέσματος, ανάλογα με το είδος της απάντησης. Τέλος, για να αντιμετωπίσουμε και τα προβλήματα ποιότητας της τελικής απάντησης (καθώς η προηγούμενη μέθοδος παρουσίασε αποκλίσεις σε δύο είδη προσεγγιστικών λύσεων που προτείνονται στο χρήστη), εισάγουμε μια τρίτη εκδοχή της μεθόδου, στην οποία υπολογίζουμε το ιστόγραμμα του υψηλότερα αποδεκτού κόμβου (σε σχέση με τους περιορισμούς που θέτει ο χρήστης) στο χρόνο εκτέλεσης. Αυτή η μέθοδος κοστίζει 0.1-0.3 δευτερόλεπτα για κάθε αίτημα ενός χρήστη, αλλά κερδίζει εξαιρετική ποιότητα τελικής λύσεις για όλα τα είδη απαντήσεων. Έτσι, μπορούμε να επιτρέψουμε στον διαχειριστή να διαπραγματεύεται την ποιότητα της λύσης, το χρόνο που θα πάρει για να την λάβει καθώς και της παραμέτρους του αιτήματος του χρήστη.

ABSTRACT IN ENGLISH

Alexandra Pilalidou, MSc, Computer Science Department, University of Ioannina Greece. July, 2010 Online Negotiation for Privacy preserving data Publishing.
Thesis Supervisor: Panos Vassiliadis.

The problem of *privacy preserving data publishing* is defined as the problem of publicly presenting a data set with the structured records around the activities or transactions of a set of persons, in order to accommodate the following two antagonistic goals: (a) allow a set of well-intended knowledge workers to execute data mining algorithms over the public data set in order to extract useful information of statistical nature for this data set, and, (b) prevent a malicious attacker to combine these publicly available data with background knowledge (in the sense of personal knowledge of the attacker, other publicly available data sets, etc) in order to link a specific person in the real world (and in particular *sensitive* information around this person) with its corresponding record in the public data set. The main technique that data curators undergo is the *anonymization* of data, which involves transforming the data (in one of many ways that the research community has come up with) before presenting them for public use. In our setting, we focus on the global recoding approach which is a method for data anonymization with (a) high utility for the data mining tools of the well-intended users, (b) faster times than the alternative methods (although not fast enough for an online environment), and, at the same time, (c) the problem of having to delete (a.k.a., suppress) outlier groups to attain an acceptable level of generalization.

In this thesis we attack the following goals, not previously explored by the research community. The first goal of this thesis is to study the interplay of suppression, generalization and privacy criterion and record how changes to one of these parameters affect the two others. The main goal, however, of this thesis is to provide the means to negotiate the configuration of the anonymization of a data set, by

- allowing a target group of known well-meaning users and the data curator who is responsible for the anonymization of data to agree *online* on (a) the level of data generalization (and thus, the incurred information loss for the well-meaning users), (b) the number of tuples that can be omitted from the published data set and (c) the privacy criterion that the data curator imposes.

Our first approach involves precomputing suitable histograms for all the different anonymization schemes that a global recoding method can follow. This allows computing exact answers extremely fact (in the order of few milliseconds). We provide both exact answers, if they exist, and suggestions for approximate answers by exploiting these histograms. However, this approach requires a pre-processing time in the orders of few dozens of minutes; whenever this is not feasible, alternative approaches must be explored. To this end, we propose a method that precomputes a small subset of the histograms in order to speed up the pre-processing time. Our experiments indicate a linear speedup along with very good or acceptable values for the quality of the proposed solutions, depending on the type of answer. Finally, to alleviate the problems of deviations from the optimal solution for two cases of approximation suggestions, we introduce a third variant, where the histogram of the top acceptable node (in terms of height constraint) is also computed at runtime. This method pays the price of 0.1-0.3 seconds to gain excellent quality of solution for all kinds of answers. This way, the data curator is equipped with alternative tools that he can use depending on the constraints in terms of user time and quality of solution.

CHAPTER 1. INTRODUCTION

It is Monday morning and the deputy minister of the Ministry of Health and Insurance arrives at his office. In the corridor, he finds three angry people quarrelling: the Chief Information Technology (CIT) officer, his legal councillor and his strategic planning advisor.

Advisor: Ah, you came! Please tell them I must have these hospital data for the new Insurance law...

Lawyer: No, you can't! It is against the law to have access to the data unless they are appropriately anonymized! Individuals must be hidden in the crowd before you can have access to their health data.

CIT: But we have anonymized the data and he doesn't like them!

Advisor: You call these 'data'? Not only did you generalize the details of the records, but you have suppressed 10% of the data set!

Minister: Is this right? Can't you give him at least the full data set, without deleting records?

CIT: We did! Twice! The first time, we used a technique called full-domain generalization and he complained the data were too much generalized...

Advisor: .. you bet they were...

CIT: and the second time, we used a more elaborate technique called local recoding, and he complained that the data were not suitable for him

Advisor: You IT people you are always giving me headaches. We have spent zillions of hours in meetings with all the 5 departments of the ministry trying to 'reconcile the warehouse dimensions' as you said, because your precious warehouse wouldn't work otherwise. And now you 're telling me that after all this effort, these 'dimensions' and their hierarchies are no good because you had to do this local recoding of yours and give me age groups of 17 – 32 that you think have any meaning to anybody!

Minister (holding the CIT's hands --who is making a threatening move-- before he punches the advisor): Isn't there any other way?

CIT: Well, his majesty refuses to take data with noise or any perturbation of values and we showed him a preview of a technique called data anatomization and he says that the utility of data is zero for him...

Minister: OK, I got it. I am a politician and I know it when I see it: you have to negotiate your demands and seek for a compromise for the antagonistic demands of information utility, hiding in the crowd, suppression, and generalization...

CIT: Yeah, right, if we could do this interactively we wouldn't be here on Monday morning shouting outside your office...

Privacy preserving data publishing is the problem of publicly presenting a data set that includes information around the activities or transactions of a set of persons, in the form of structured records in order to accommodate the following two antagonistic goals: (a) allow a set of well-intended knowledge workers to execute data mining algorithms in order to extract useful information of statistical nature around the data set, and, (b) prevent a malicious attacker to combine these publicly available data with background knowledge (in the sense of personal knowledge of the attacker, other publicly available data sets, etc) in order to link a specific person in the real world (and in particular *sensitive* information around this person) with its corresponding record in the public data set. The main technique that data curators undergo in order to process the available data before making them public is the *anonymization* of data, which involves transforming the data (in one of various ways that the research community has come up with) before presenting them.

To give a simple example, assume that the data curator of a hospital wants to make patient records publicly available to knowledge workers without allowing malicious users understand which records corresponds to which person in the real world. In the case of the patient records of our example, the disease, symptoms and treatment of each patient are examples of such sensitive values. To achieve that, a set of *identifier* attributes of data are removed (in the case of patients' example, this would be the name, SSN, tax-agency-id, or other similar attributes). However, this is not enough: as the bibliography has characteristically shown for the case of Massachusetts' governor

[Swee02a] it is still possible to identify to whom a record corresponds via a set of so-called *quasi identifier* attributes (for example, here: zipcode, age, sex) whose combination might uniquely characterize a person. In our example, a patient's neighbor who knows (a) the zip code, sex and age of a patient, and (b)the fact that the patient was hospitalized on a specific date, can reason on the patient's disease if there are no other patients with similar characteristics. To attack this vulnerability, the research community has come up with a variety of techniques that aim to abstract the detailed values of the original records with more *generalized* values in the published data set: so, instead of publishing the exact zip code 45110, a generalized version of it might be published: 4511*, or 451**. Similarly, instead of publishing that the age attribute has the value of 35 for a certain record, one might publish that the age belongs to the range [31, 40]. A published data set is *k-anonymous* if every record in the published data shares the same quasi identifier value with at least k-1 other tuples. Sometimes, a generalization scheme (i.e., a decision on the level of abstraction for each of the quasi-identifiers) produces nice groups with the exception of some outlier groups that violate the k-anonymity criterion. Some of the published approaches allow the deletion of these tuples, which is known as suppression in the literature. As we shall see in the sequel, the management of suppression is a non-trivial problem for the data curator.

The research community has provided several methods and statistical tests to allow the effective publishing of private data. One line of research deals with the *privacy criterion*: any privacy criterion (like the abovementioned criterion of k-anonymity) suffers from vulnerabilities of statistical nature; therefore different privacy criteria have been developed over time, each covering weaknesses of the previous (with the computational complexity and the possibility of suppressing the whole data set in the end being the main drawbacks of more and more sophisticated privacy criteria). Another line of research deals with the nature of anonymization process: instead of generalizing the data set it is also possible to introduce noise or destroy the linkage of quasi-identifiers to sensitive values. However, one of the criticisms against these approaches is that although each of these methods seems to perform quite well in terms of the privacy offered for the individuals whose records are in the public data set, it also appears to annoy the knowledge workers since the "world's truth" presented by the data set is either false, or destroyed in terms of utility. Even the area of generalization has several possible approaches within it: for example, why do we have to generalize uniformly all postal codes? If a geographical region is densely represented in the data set, then the need to generalize it at the same level of abstraction with a region which is sparsely represented is not directly obvious (since the former can easily reach groups of the desired privacy criterion at lower levels of abstraction than the latter). Therefore, approaches that customize the partition of data to groups in non-uniform ways (also known as local recoding of values, as opposed to the global recoding of values alternatively suggested) have been proposed in the literature. Despite their obvious advantage, which is the minimization (or actually, the elimination) of suppression, it has also been argued that these approaches are slow, make it extremely difficult for the data mining tools to extract useful knowledge and present the users with unnatural groupings of data [FWCY10].

Despite all this activity, there are several issues not covered by the research community so far, that we try to address in this thesis. To the best of our knowledge, this is the first time that these issues are explored in a systematic way. *The first problem involves the systematic study of the relationship between suppression, generalization, and privacy criterion.* In other words, what is the amount of generalization that appears to be necessary before we restrict suppression to tolerable ranges? What is the role of the value of the privacy criterion in this relationship? *A second problem that this thesis addresses is the proposal of efficient ways that allow the user achieve an anonymous data set with constraints over the generalization height, the amount of suppression and the tunable value of the privacy criterion. A third, related problem involves the ability to provide suggestions to the user that are close to his original desideratum around generalization, suppression and privacy.* The desideratum is that the user negotiates interactively with an anonymization system the properties of an anonymized data set. If, for example, the user sets a suppression threshold too low for the anonymization to attain the privacy criterion that he also sets, then the system should ideally respond very quickly with a negative answer to the user, along with a set of proposals on what possible generalizations, close to the one that he originally submitted, are attainable with the specific data set. This practically requires the ability to provide answers to the user in user time.

Problem statement. The main goal of this paper is to provide the means to negotiate the configuration of the anonymization of a data set, by allowing a target group of known well-meaning users and the data curator who is responsible for the anonymization of data to agree on (a) the level of data generalization (and thus, the incurred information loss for the well-meaning users), (b) the number of tuples that can be omitted from the published data set and (c) the privacy criterion that the data curator imposes.

We make the following **assumptions**:

- We assume that the end users require that data are published with respect to a set of generalization hierarchies whose members and structure are predetermined. To put this in context, we assume that the users have been working with the dimensions of a data warehouse for some time and have a strong point of view on how they want information presented to them. Therefore, they are quite reluctant to work with automatically computed intervals of values that are typically produced by local or multidimensional recoding methods.
- Moreover, we assume that the data curator has a range of acceptable values for the privacy preservation criterion (e.g., for the parameter *l* of l-diversity) and, despite the fact that he starts with a preferred value, he does not set a strict constraint on a specific value.
- Another assumption has to do with the possibility of omitting ("suppressing") tuples from the published data set. The omission of tuples clearly results in (sometimes high) information loss; however, sometimes, removing a set of outlier tuples can allow the generalization of the data set to a much lower level of abstraction, thus resulting to a published data set that is more rich in information rather than if the tuples were retained. We allow, thus, the suppression of tuples; however we impose the reasonable constraint that a maximum number of suppressed tuples is acceptable by the end users.

Furthermore, we operate on the basis of the following **soft-constraints**:

- Among several possible anonymization schemes for the same data sets, we need to discover the one that best fits the user's needs. Typically, a decision criterion, in the form of a *utility function* is employed for the assessment of the quality of a candidate solution. In the rest of our deliberations, we use a simple decision criterion by default, concerning the height of a possible solution in the anonymization hierarchies, since it is very intuitive for the user and possesses nice monotonicity properties.

- Finally, we pose as a soft-constraint the desideratum of a non-strict privacy criterion. We want our method of privacy preservation to be pluggable to the proposed framework and retain the possibility of choosing among alternative methods for privacy preservation (e.g., k-anonymity, l-diversity, t-closeness, X, Y anonymity etc). In our deliberations we will focus on two practically attainable criteria, specifically k-anonymity and l-diversity; however other criteria are also applicable to our method.

In summary, we can state the problem we are attacking as follows:

Given

(a) a data set *T*, comprising an identifier attribute *ID*, a set of quasi-identifier attributes **QI** = {A_1, ..., A_n}, and a sensitive attribute *S*,

(b) a set of generalization hierarchies **H** = {$\mathbf{h}_1$, ..., $\mathbf{h}_n$}, one for each quasi-identifier attribute,

(c) a privacy constraint (e.g., *k*-anonymity, *l*-diversity, ...),

(d) fixed constraints for (d1) the maximum height per attribute that the anonymization method can attain **h** = [h_1^c, ..., h_n^c], (d2) the lowest value for the privacy constraint (e.g., *k* for *k*-anonymity) and (d3) the maximum number of suppressed tuples that the user is willing to tolerate *MaxSupp*,

(e) a quality criterion function *QoS*() for the assessment of the best possible anonymization when more than one answers are available (e.g., the solution with the lowest height, and possibly the less suppressed tuples, or maximum discernibility, as another example).

Produce

(i) An anonymized data set *T** such that

- *T** is a generalization of *T*,
- *T** fulfils the abovementioned privacy constraints (d1) – (d3), and,
- *T** minimizes the quality criterion function *QoS*(*T**),

if such a *T** can be attained,

or,

(ii) A set of alternative generalizations that are also generalizations of T and each of them minimizes the deviation for one of the parameters of the problem, specifically, (a) the acceptable generalization heights, (b) the minimum acceptable value for the privacy constraint and (c) the number of suppressed tuples.

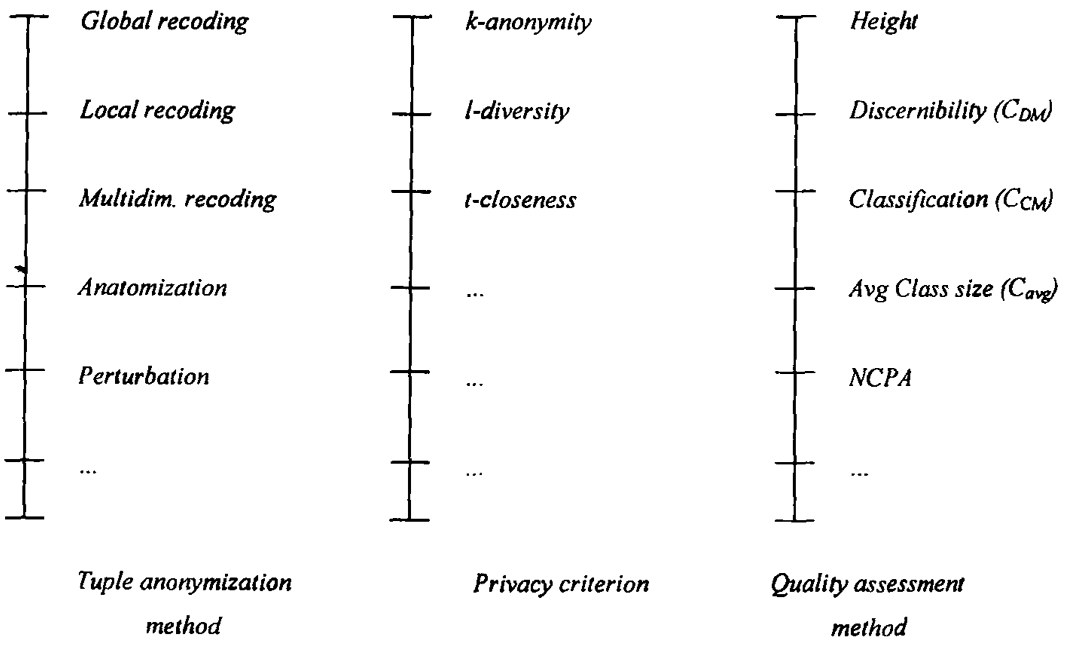


Figure 1.1 Problem Parameters

The possible values for different parameters of the problem are depicted in Figure 1.1. The anonymization method can be any of global / local /multidimensional recoding [LeDR05], [LeDR06], [Xu+06], [LWFP08], [GhKM09], tuple perturbation [AgST05], [ZKSY07], anonymization [XiTa06] or other. The privacy criterion can be any of k-anonymity [Swee02a], l-diversity (in any of its forms) [MaGK06], t-closeness [LiLV07], or other. The function that assesses the quality (or penalty) of a candidate solution can be the height of the solution [Sama01], the discernibility metric [BaAg05], the average class size [LeDR06], or other.

- **Our approach.** In our case, we start with a simple setting, comprising k-anonymity and l-diversity, global generalization and solution height as the choices of preference. The first method proposed in this thesis involves precomputing statistical information for several possible generalization scheme. A generalization scheme is determined by deciding the level of generalization for every quasi-identifier – in other words a generalization scheme is a vector characterizing every quasi-identifier with its level of generalization. To efficiently compute the amount of suppression for a given pair of (i) value for the privacy criterion and (ii) a generalization scheme, we resort to the precalculation of a histogram per generalization scheme that allow us to calculate the necessary statistical information. For example, in the case of k-anonymity we group the data by the quasi identifier set of attributes in their generalized form and we count how many groups have size 1, 2, ... etc. So, given a specific value of k, we can compute how many tuples will be suppressed for any generalization scheme. Similarly, in the case of l-diversity, we count the number of different sensitive values per group along with the size of the group per group.

We organize generalization schemes in a lattice. A node *v* is lower than a node *u* in the lattice if *u* has at least one level of generalization higher than *v* for a certain quasi-identifier and the rest of the quasi-identifiers in higher or equal levels. Once the histogram is computed for every node in the lattice, the main algorithm checks whether there exists a possible solution to the abovementioned problem that satisfies all criteria. This is performed by first checking the solutions in the *top-acceptable-node* v_{max} defined with generalization levels [h_1, ..., h_n]. If a solution exists then we exploit a simple monotonicity property and look for possible answers in quasi identifiers with less or equal generalization levels than the ones of the top acceptable node. In the case that no solution exists in the top acceptable node, the algorithm provides the user with 3 complementary suggestions as answers:

- The first suggested alternative satisfies *k* and **h** but not *MaxSupp*. In fact, we search the space under the top acceptable node and provide the solution with the minimum number of suppressed tuples. In typical situations, we can guarantee that the answer is already in found in the top acceptable node and by exploiting the original search of the top acceptable node, we can provide the answer immediately.

- The second suggested alternative is a solution that finds the maximum possible *k* for which **h** and *MaxSupp* are respected for the quasi identifiers of the top acceptable node. Again, this is an answer that can be provided immediately by exploiting the search of the top acceptable node.
- Finally, the third alternative is a solution that satisfies *k* and *MaxSupp* but violates **h**. This means that we have to explore the space of quasi identifiers that are found in generalization levels higher or equal than the top acceptable node. We exploit some monotonicity properties already discussed early in the literature [Sama01] to avoid unnecessary checks and utilize a binary search exploration of heights on the lattice.

The proposed method is guaranteed to provide the best possible answers for the given user requests. Our experiments indicate that this is performed in less than 10 milliseconds for typical data sets used in the research literature.

However, the method comes at a price, and specifically, at the price of precomputing the histograms for all the nodes of the lattice. This precomputation requires several minutes (e.g., our experiments gave 20-40 minutes for the largest quasi-identifier sets). If one is to avoid the cost of full precomputation, we need to devise an alternative approach. So, in this thesis, we explore a second approach that tries to precompute a small subset of the lattice's nodes with their histogram. The goal is to carefully select the generated nodes in order to (a) minimize the deviation from the optimal solution and (b) precompute the necessary subset of the lattice in times that are tolerable by the users. Our approach is based on the ranking of generalization levels with respect to their grouping power (since, the larger the groups, the less the suppression). Then, we try to rank the combinations of levels for all the possible generalization schemes and pick a fixed subset of them (e.g., 5%). Our experiments demonstrate a linear speedup of the precomputation time with the approximation factor, very good performance for the provision of exact answers and level relaxations, as well as certain deviations in terms of the approximate generalization heights and suppressions.

Finally, by observing that the two out of the three alternatives suggested in the absence of an exact answer are due to the top-acceptable node, we propose a third method that computes the histogram of this node at runtime. Our experiments demonstrate that the time penalty for this extra computation is in the order of 0.1 – 0.3 sec and the two relaxations that suffered in the previous approach demonstrated an identical behavior to the case of the full lattice; therefore, if this time overhead can be tolerated in terms of user time (and for the case of our experiments we believe it does), then the quality of solution improves drastically.

- **Roadmap.** In Chapter 2, we discuss the fundamental concepts of the problem under investigation. In Chapter 3, we explore the interplay of the problem's parameters, specifically, the size of the quasi-identifier set, and the values for the privacy criterion and the acceptable suppression. In Chapter 4, we discuss the proposed method with a full precomputation of the lattice of generalization schemes. In Chapter 5, we discuss alternatives to this full precomputation. In Chapter 6, we discuss related work. Finally, in Chapter 7, we conclude with our findings and present insights for future work.

CHAPTER 2. FUNDAMENTAL CONCEPTS AND TERMINOLOGY

2.1. Motivating Example

Assume that a trusted data curator has collected the microdata table displayed in Figure 2.1. The microdata comprise (a) an identifier attribute, *Name*, (b) a set of quasi-identifier attributes, specifically, *Age*, *Work Class*, and *Education*, and (c) a sensitive attribute, (*Working*) *Hours per Week.* Each attribute is accompanied by value hierarchies, pretty much in the way OLAP dimensions are organized in value hierarchies. So, for example, the *Education* of a person who has attended school till the 11th grade, is characterized with respect to different levels of abstraction as (a) Detailed: 11th-grade, (b) Level 1: Senior secondary, (c) Level 2: Secondary, and (d) Level 3: Without Post Secondary. As another example, *Age* can be organized in terms of years, 5-year intervals, 10-year intervals, etc. In Figure 3.1, the hierarchies for the attributes *Work Class* and *Education* can be inspected in detail.

We want to publish the data under the following setting:

(a) every tuple belongs to a group of tuples with the same quasi identifiers, with size at least 3 (i.e., the privacy constraint is k-anonymity, with $k = 3$)

(b) no tuples are suppressed (i.e., *MaxSupp* =0)

(c) *Age* and *Work Class* can be generalized at most 1 level, whereas *Education* can be generalized at most 3 levels up (i.e., **h**= [1, 1, 3]).

Name	Age	Work_class	Education	Hours/week
Thales	39	Private	Hs-grad	40
Anaximander	38	Private	Hs-grad	50
Anaximenes	37	Private	Hs-grad	40
Pythagoras	38	Private	11th	45
Gorgias	28	Loc-gov	Bachelors	30
Heraclitus	31	Federal-gov	Master	50
Empedocles	30	State-gov	Bachelors	60
Leucippus	32	Self-emp-not-inc	Bachelors	50
Democritus	35	Self-emp-inc	Prof-school	54
Protagoras	33	Self-emp-inc	Assoc-acd	40

Figure 2.1 Microdata table (Based on Adult data set)

As one can see in Figure 2.2 this setting is feasible. The microdata table is partitioned in three groups, each having at least 3 tuples. No tuples are suppressed and the generalization is respected in all three quasi identifiers. The color and format of the tuples in Figure 2.2 suggests the group to which they belong to. The identifier attribute *Name* is not published and presented here for intuition reasons only.

Name
Thales
Anaximander
Anaximenes
Pythagoras
Gorgias
Heraclitus
Empedocles
Leucippus
Democritus
Protagoras

Age	Work_class	Education	Hours/week
37-41	**Private**	**Without-post-secondary**	**40**
37-41	**Private**	**Without-post-secondary**	**50**
37-41	**Private**	**Without-post-secondary**	**40**
37-41	**Private**	**Without-post-secondary**	**45**
27-31	Gov	Post-secondary	30
27-31	Gov	Post-secondary	50
27-31	Gov	Post-secondary	60
32-36	*Self-emp*	*Post-secondary*	*50*
32-36	*Self-emp*	*Post-secondary*	*54*
32-36	*Self-emp*	*Post-secondary*	*40*

Figure 2.2 Generalized data set (3-anonymous, no suppression, h=[1,1,3]).

Assume now that we want to achieve a 4-anonymous generalization of the microdata, still retaining the constraints for no suppression and generalization heights (i.e., *MaxSupp* = 0 and **h**=[1,1,3]). One can see that we cannot attain such a setting (only one group has size 4, the rest comprise of only three tuples). Then, we need to perform some relaxation to our constraints. Several such relaxations can be suggested to the user:

- The first suggested alternative satisfies *k* and **h** but not *MaxSupp*. We see that a possible solution suppresses all the groups with less than 4 tuples, thus removing 6 tuples.. Then, the resulting data set is depicted in Figure 2.3.

Name	Age	Work_class	Education	Hours/week
Thales	37-41	Private	Without-post-secondary	40
Anaximander	37-41	Private	Without-post-secondary	50
Anaximenes	37-41	Private	Without-post-secondary	40
Pythagoras	37-41	Private	Without-post-secondary	45

Figure 2.3 Generalized data set with suppression relaxed (4-anonymous, h=[1,1,3], but 6 tuples suppressed).

- The second suggested alternative is a solution that finds the maximum possible *k* for which **h** and *MaxSupp* are respected for the quasi identifiers of the top acceptable node. Clearly this is the generalization of Figure 2.1 (since it only suffices to reduce *k* = 4 by one to achieve it).
- Finally, the third alternative that can be suggested is a solution that satisfies *k* and *MaxSupp* but violates **h**. We can try to ascend the hierarchy for every quasi identifier attribute by one level, until the desired suppression is achieved. So, we ascend attribute *Age* by one level and present ages in intervals of 10 years. However, we still have the same three groups as in Figure 2.2 (albeit, with different values in the age field). Then, we ascend attribute *Work Class* by one level, to a level that comprises two values only worked and never worked). These two transitions manage to merge the second and third group of Figure 2.2 into a single group comprising 6 tuples. This way, both constraints regarding group size and suppression, *k* = 4 and *MaxSupp* = 0, are supported and the result is the one depicted in Figure 2.4.

Name	Age	Work_class	Education	Hours/week
Thales	37-46	Worked	Without-post-secondary	40
Anaximander	37-46	Worked	Without-post-secondary	50
Anaximenes	37-46	Worked	Without-post-secondary	40
Pythagoras	37-46	Worked	Without-post-secondary	45
Gorgias	27-36	Worked	Post-secondary	30
Heraclitus	27-36	Worked	Post-secondary	50
Empedocles	27-36	Worked	Post-secondary	60
Leucippus	27-36	Worked	Post-secondary	50
Democritus	27-36	Worked	Post-secondary	54
Protagoras	27-36	Worked	Post-secondary	40

Figure 2.4 Generalized data set with generalization height relaxed (4-anonymous, no suppression, but h=[2,2,3]).

2.2. Background and Terminology

In this section, we will formally introduce the fundamental concepts around the issues of anonymization that we will address in this paper. We distill several well-known concepts in the related literature; consequently, the interested reader can also refer to [Sama01, LeDR05, MaGK06, LWFP08] for alternative presentations of these concepts.

We start by assuming a *microdata* relation *R* containing all the detailed information. We have three categories of users. First, we assume there is a trusted data curator with full access to the detailed information whose job description includes the publishing of data without sacrificing the privacy of the persons to whom the data correspond. We assume that the data curator is trusted. We also have a set of *well-meaning analysts* who apply data mining algorithms over the published data whose aim is to find statistically important information about the data set, but not anything in particular for specific individuals. Finally, we also have a set of *attackers* whose aim is to discover the correct values for one or more persons in the real world, by exploiting any published information available (and not necessarily the published version of *R*).

The attributes of *R* can be divided in the following categories:

- Identifiers: these are attributes allow anybody with access to the microdata of R to relate a tuple of *R* with a person in the real world. For example, a person's name, or SSN belong to the identifier class of attributes. When publishing data, identifiers are removed from the published data set.
- Quasi-identifiers: A set of attributes is called a Quasi-Identifier Set if the combination of these attributes allows a person with access to the published data set to relate a tuple of this data set to its hidden identifier (and consequently, to a person in the real world). In the example of Fig. 2.1 assuming that the attacker knows that Heraclitus is working for the federal government and has a bachelors degree, even if the name is projected out of the published relation, a quick glance at the remaining columns quickly reveals that there is only one person with the characteristics of Heraclitus in the data set. Therefore, even if the name is removed, the combination of *Work Class* and *Education* is sufficient for an attacker to relate the respective tuple to the hidden identifier (i.e., attribute *Name*). The set of quasi-identifier attributes of a relation will frequently be referred to as *QI* as a shorthand. An attribute that is member of the quasi-identifier set is called a quasi-identifier.
- Sensitive attributes: A sensitive attribute is an attribute whose value must not be linked to a hidden identifier value by an attacker. The core of the private data publishing problem is to alter the original data set in such a way that the published data set restricts the probability of relating the published value of a sensitive attribute to the hidden identifier of a tuple. For example, in a patients' data set, the name of the patient and disease that she suffered must not be linked by an attacker. In our example, it is the task of the data curator to prevent an attacker from relating a (hidden) *Name* identifier (e.g., Heraclitus) to the value of *Hours per Week* that he works (here: 50).
- Indifferent attributes: these are any other attributes of the data set that we do not care if they can be linked to a hidden identifier.

As typically happens in the literature, we will assume that there is one sensitive attribute in the microdata and that no indifferent attributes are present in the data set, unless this is explicitly stated. So, without loss of generality, we assume that *R* is defined as R (A_{ID}, A_1, A_2, ..., A_n, S), where A_{ID} is an identifier, A_1, A_2, ..., A_n is the quasi-identifier set and S is the sensitive value.

The quasi identifier attributes are accompanied by value hierarchies in a way that resembles a lot the way OLAP dimensions organize their values in hierarchies. We assume the following setting for quasi-identifier attributes and their domains.

- Every attribute A is accompanied by a domain of values, *dom(A)* that is isomorphic to the integers. Typically, attributes can be either nominal or arithmetical. The isomorphism to the integers is not obvious for the nominal values; however, an artificial ordering can be imposed to the domain of such attributes (especially, if, as typically happens, the microdata table has a foreign key to a lookup table for the quasi identifier).
- Every quasi-identifier attribute is part of hierarchy of attributes. A hierarchy of attributes H is a finite list of attributes, whose first member is the most detailed level of values (the one that belongs to the microdata table too) and the last member is the level H.All: $H=\{A_0, A_1, A_2, \ldots, A_n, H.\text{All}\}$. The attributes that participate in a hierarchy are called *anonymization levels*, or simply *levels* of the hierarchy (in correspondence to the *aggregation levels* in an OLAP context). The higher an attribute is in the hierarchy, the coarser the level of semantic abstraction its values have. The level All stands for complete anonymization of the values for this attribute; to this end, its only member is a single value, *. For example, the quasi-identifier attribute Age can belong to a hierarchy with values at the year level, 5-year intervals, and 10-year intervals: $H_{\text{age}} = \{Age_{\text{year}}, Age_{\text{5-year}}, Age_{\text{10-year}}, Age.\text{All}\}$. Whenever an attribute A_{high} is at a higher level in a hierarchy than an attribute A_{low}, we denote this by the notation $A_{\text{low}} \rightarrow A_{\text{high}}$. We will frequently reuse terminology from the domain of OLAP and refer to a hierarchy of attributes as a *dimension*, whose attributes will also be called *levels* (of detail).
- We assume a full mapping between the domains of the attributes of a hierarchy, denoted as $\underset{A_l}{\overset{A_h}{anc}}$. Formally, given two attributes A_{low} and A_{high}, $A_{\text{low}} \rightarrow A_{\text{high}}$, $v_h = \underset{A_l}{\overset{A_h}{anc}}(v_l)$ is a total function $\underset{A_l}{\overset{A_h}{anc}}$: $dom(A_{\text{low}}) \rightarrow dom(A_{\text{high}})$ returning a value v_h at a coarser level for a value v_l at a lower level. In other words, for every detailed value (e.g., Age 37 years at the detailed level) there is a single value at the coarser level (e.g., the interval [31-40] years) to which it corresponds. We reuse the

notation $v_l \rightarrow v_h$ for the values of the respective domains. The $anc_{A_l}^{A_h}$ function is defined as the identity function if $A_{low} \equiv A_{high}$.

- An extra well-formedness constraint involves the composition of ancestor functions. For any values of any three levels A_1, A_2, A_3, such that $A_1 \rightarrow A_2 \rightarrow A_3$, the following property must hold: if $v_2 = anc_{A_1}^{A_2}(v_1)$ and $v_3 = anc_{A_2}^{A_3}(v_2)$, then $v_3 = anc_{A_1}^{A_3}(v_1)$ too.
- We call a hierarchy *ragged* if the mapping of values in not full for all the domains of all attributes. For example, observe the value 'Without pay' in the third level of the hierarchy for the quasi-identifier *Work class*. The value 'Without pay' does not have any descendants mapped to it at the levels *L*0 and *L*1, thus violating the definition of a hierarchy. Ragged hierarchies are easy to compensate by adding artificial representatives of coarse values at the detailed levels where such representatives are missing. For example, in the case of the value 'Without pay' in L2, we introduce two artificial values 'W/O pay_L1' at level *L*1 and 'W/O pay_L1' at level *L0*, and update the ancestor function appropriately to incorporate all these three values. Therefore, in the sequel, we do not consider ragged hierarchies at all.

A **full domain**, or **global**, **generalization** of a relation $R(A_{ID}, A_1, A_2, ..., A_n, S)$ is a new relation P that is produced by (a) the projection of the non-identifier attributes and (b) the replacement of the values of a quasi-identifier attribute with their respective ancestor values on the basis of the hierarchies previously defined. Naturally, the ancestor function that is employed for an attribute can be the identity function.

Formally, we say that a relation R $(A_{ID}, A_1, A_2, ..., A_n, S)$ is **fully generalized** to a relation P $(Q_1, Q_2, ..., Q_n, S)$, or, equivalently, that P is a **full domain generalization**, or, **global generalization** of R, if

(a) at the schema level, $Q_i = anc(A_i)$, for all $i = 1, .., n$, and,

(b) for every tuple t in R, we introduce a tuple t' in P, such that $t[S] = t'[S]$, and, for every attribute Ai of R, the value $t[A_i]$ is replaced by a value $t'[Q_i]$.

A **generalization scheme** of a relation R (A_{ID}, A_1, A_2, ..., A_n, S) is a set of quasi identifiers $QI= \{Q_1, Q_2, ..., Q_n\}$ that produce a full domain generalization P (Q_1, Q_2, ..., Q_n, S) of R. Given a specific generalization scheme as above, we refer to the level Q_i as the *generalization level* of attribute A_i.

k-anonymity [MaGK06]. A relation T (be it microdata or a generalized relation) is said to be k-anonymous with respect to a set of (generalized or not) quasi-identifier attributes $QI= \{Q_1, Q_2, ..., Q_n\}$, if every tuple t in T, there exist at least k-1 other tuples t_{i1}, t_{i2} , . . . , t_{ik-1} in T such that $t[Q] = t_{i1}[Q] = t_{i2}[Q] = \cdot\cdot\cdot = t_{ik-1}[Q]$ for all quasi-identifiers Q in QI.

Blocks (equivalence classes). We will refer to a set of tuples of a relation T under a generalization scheme QI with the same values of quasi identifiers (again, independently of their level of generalization) as a block or equivalence class for relation T and its generalization scheme.

Observe that a full domain generalization produces a partition of the published relation T to blocks/partitions/equivalence classes on the basis of the generalization scheme. In other words, all tuples belonging to a block form an equivalence class. By definition, these partitions are disjoint, and then, T is the union of these disjoint partitions.

Clearly, k-anonymity is the first attempt to hide individual tuples in the crowd. A k-anonymous generalization protects a tuple from an attacker by placing it in a block of at least k tuples with the same quasi identifier values. This way, if an attacker knows the quasi-identifier values for a person in a real world, the tuple that corresponds to the victim is 'hidden' in the crowd of its respective block and it is harder for an attacker to relate the hidden identifier to the correct sensitive value via the quasi identifier. There are several weaknesses of k-anonymity (see, for example [MaGK06]) and so several extensions are constantly being developed by the research community.

In the context of this paper, we restrict ourselves to the simplest –yet quite powerful- extension of simple l-diversity [MaGK06] that tries to address the problem on non-diversity of the sensitive values within a block: if all (or a significant fraction of) the tuples of a block have the same sensitive value, then the block gives away (with certainty or high probability) the sensitive value of the victim. So, the sensitive values of the tuples of the same group must be quite diverse ("well represented" in the [MaGK06] terminology). The simplest (and most popular) way to do this is to ensure that every block possesses at least *l* distinct sensitive values.

Simple l-diversity. A generalization T satisfies simple l-diversity, if in every block *q*, no more than $\frac{1}{l}$ of the tuples have the same sensitive value.

2.3. The annotated lattice of generalization schemes

2.3.1. The lattice of generalization schemes

The possible generalization schemes that can occur via a combination of anonymization levels for different quasi identifiers can be organized in a lattice. In this section, we will formally introduce the lattice; discuss how it can be produced and what its properties are. A first discussion of the lattice is in [Sama01, Incognito].

Lemma. A hierarchy forms a total order at the intentional level and a partial order at the extensional level.

Proof.

At the intentional level, by definition we assume that the anonymization levels of a hierarchy form a line. Thus, for any two levels A_x and A_y, one must precede the other (either $A_x \rightarrow A_y$, or $A_y \rightarrow A_x$) with the → function being the ordering function of the total order.

At the intentional level, it is easy to show that the values of a hierarchy form a tree: there is a single value (*) at the top level of the hierarchy, and every value has a single ancestor value at the preceding anonymization level (remember, the *anc* function is both total and a function). Thus, the resulting hierarchy of values can form a tree with the values as nodes and an edge between two values if they belong to consecutive levels and they are related via an *anc* function.

A partial order for a set of values P and an ordering function $\leq$, imposes three constraints: reflexivity ($x \leq x$), antisymmetry ($x \leq y$ and $y \leq x$ imply $x \equiv y$) and transitivity ($x \leq y$ and $y \leq z$ imply $x \leq z$). Assuming the set of all the values of the union of the domains of the attributes of a hierarchy as the set P and the *anc* function as the ordering function, we can conclude that all three properties hold. QED

Definition. Given a set of hierarchies $H=[H_1, \ldots, H_n]$ that constitute a set of quasi identifier dimensions, the **anonymization lattice** L is the Cartesian product of the hierarchies at the intentional level.

Remember that given a set of ordered sets $P_1, \ldots, P_n$ their Cartesian product $P = P_1 \times \ldots \times P_n$ is also an ordered set with the following constraint:

$(x_1, \ldots, x_n) \leq (y_1, \ldots, y_n) \Leftrightarrow$ for each i, $x_i \leq y_i$ in Pi

In other words, every member of the Cartesian product P is annotated by one level per quasi-identifier dimension and a member x follows a member y if all the individual levels of x are lower or equal to the respective levels of y, for all the quasi identifier dimensions.

Take for example the hierarchies for the quasi-identifier set [*Age*, *WorkClass*, *Race*] as depicted in Figure 3.1. We will assume that Age has five levels of anonymization $\{A_0, A_1, A_2, A_3, A_4 \equiv A.\text{All}\}$, Workclass has four levels $\{W_0, W_1, W_2, W_3 \equiv W.\text{All}\}$, and Race has 3 levels of anonymization, too $\{R_0, R_1, R_2 \equiv R.\text{All}\}$. In all our deliberations in the sequel, we will assume that the order of quasi-identifiers is fixed; for example, in this case, we will always list the attribute *Age* first, *Workclass* second and *Race* third. Consequently, when we refer to the node with levels *A*2, *W*3, *R*0 we can refer to it as 230 for shorthand. The lattice for the quasi identifier set is depicted in Figure 2.5.

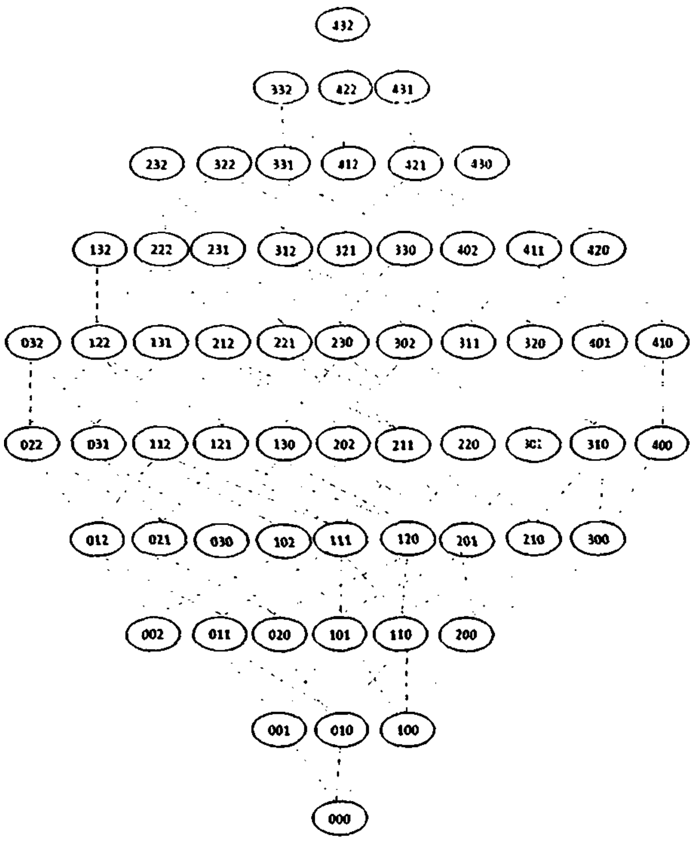


Figure 2.5 A lattice for the three quasi identifiers of the reference example Age, Work class and Race.

So far, we refer to the result of the Cartesian product of the quasi identifiers as a lattice, but we have not proved that it is indeed a lattice.

Lemma. The ordered set that results as a Cartesian product $P = H_1 \times \ldots \times H_n$ over a set of anonymization hierarchies is a lattice.

Proof. For an ordered set to be a lattice, two constraints must hold, for any two members of the set x and y:

- x and y have a supremum or join or least upper bound (i.e., there always exists a member z such that both x≤z and y≤z) – we denote this as x∨y
- x and y have a infimum or meet or greatest lower bound (i.e., there always exists a member z such that both z≤x and z≤y) – we denote this as x∧y

It is easy to see that the Cartesian Product P has a unique bottom element (typically denoted as ⊥) which is $H_1.L_0$, $H_2.L_0$, …, $H_{n-1}.L_0$, $H_n.L_0$ and a unique top element (typically denoted as T) which is $H_1.L_{all}$, $H_2.L_{all}$, …, $H_{n-1}.L_{all}$, $H_n.L_{all}$. Therefore, any two members of the lattice will at least have these two as supremum and infimum (although not necessarily them). QED.

Discussion for the extensional level. Similarly to the intentional level, one can explore the Cartesian product at the extensional level. We will not delve in the particularities of this aspect, since we will not use the Cartesian product of the hierarchies at the extensional level. Notice however that the result is not a lattice (in contrast to the intentional level having a single member at the bottom of the list, the extensional level has several members at the bottom of the tree; thus, the resulting partial order does not have a unique bottom element).

How big is the lattice? Assume *n* dimensions [*D*1, ..., *D*n] , each with levels *levels*(*D*i) levels (including the top and bottom elements). The total number of nodes in the lattice is

$$|L| = \text{levels}(D1) \text{ x } \text{levels}(D2) \text{ x } \ldots \text{ x } \text{levels}(Dn)$$

Assuming λ levels per dimension on average, this quantity is approximated by λ^n.

2.3.2. Annotation of the Lattice with histograms

Each node of the lattice corresponds to a generalization scheme. Thus, it can be annotated with information concerning the generalization scheme, the anonymization method, the number of suppressed tuples and other information related to the status of the generalization scheme represented by the node.

KA-histogram. The k-anonymity histogram for a generalization scheme $QI = \{Q_1, Q_2, \ldots, Q_n\}$ over an original microdata relation *R* is a finite list of pairs $KA = [p_1, p_2, \ldots, p_m]$ of the form *p* (*size*, *blockCount*) computed as follows:

1. The original microdata relation *R* is generalized according to *QI* and its accompanying hierarchies to a generalized relation *T*
2. We compute all the equivalence classes of *T* according to *QI* and count their sizes in terms of tuples (to be reused in the histogram as the attribute *size*)
3. For every possible size that appears, we count how many blocks (*blockCount*) are of this size. The result of this is a set of pairs of the form (*size*, *blockCount*).

Take for example the microdata table *R* of the reference example (depicted in Fig. 2.1) and its generalization *T* according to the generalization scheme *QI*=[*A*.L1, *W*.L1, *E*.L3] (depicted in Fig. 2.2). We observe that there are two blocks of size 3 and one block of size 4. The resulting KA-histogram for *T* is depicted in Figure 2.6.

size	*blockCount*
3	2
4	1

Figure 2.6 KA-histogram

Observe that the histogram does not trace which blocks are formed (although each pair can be annotated with the pairs that correspond to it). However, the histogram allows us to quickly compute the relationship of privacy to suppression. For example, given the histogram of Figure 2.6, if one wants to impose a constraint of 4-anonymity, then 6 tuples (2 groups of size 3) have to be suppressed for the corresponding generalization scheme.

Similarly to the histogram for k-anonymity one can compute the respective histogram for simple l-diversity by counting the number of distinct sensitive values that appear in a group.

SLD-histogram. The simple l-diversity histogram for a generalization scheme *QI*= $\{Q_1, Q_2, ..., Q_n\}$ over an original microdata relation *R* is a finite list of triplets *SLD*=$[p_1, p_2, \ldots, p_m]$ of the form *p*(*distinctSCount*, *blockCount*, *sumTupleCount*) computed as follows:

1. The original microdata relation *R* is generalized according to *QI* and its accompanying hierarchies to a generalized relation *T*
2. We compute all the equivalence classes of *T* according to *QI* and count the number of distinct values in the sensitive attribute within each equivalence class (to be reused in the histogram as the attribute *distinctSCount*) as well as the number of tuples for each equivalence class
3. For every possible *distinctSCount* that appears, we count how many blocks (*blockCount*) are of this size as well as the overall number of tuples that

belong to these blocks (*sumTupleCount*). The result of this is a set of triplets of the form (*distinctSCount*, *blockCount*, *sumTupleCount*).

Again, take for example the microdata table *R* of the reference example (depicted in Figure 2.1) and its generalization *T* according to the generalization scheme *QI*=[*A*.L2, *W*.L2, *E*.L3] (depicted in Figure 2.4). We observe that there are two blocks, the first having three distinct sensitive values among its four tuples and the second having five distinct values among its six tuples. If we had more than one blocks with the same *distinctSCount* value, we would sum the number of tuples that belong to each of them and obtain the overall *sumTupleCount* for this value of *distinctSCount*. The resulting SLD-histogram for *T* is depicted in Figure 2.7

distinctSCount	*blockCount*	*sumTupleCount*
3	1	4
5	1	6

Figure 2.7 SLD-histogram

Cumulative KA-histogram. Apart from the simple KA-histogram, a very convenient tool that we will employ when relating suppression with privacy is the cumulative KA histogram, which, for every size k of the KA histogram measures the number of tuples in groups with smaller size than k.

$cumKA(k) = \Sigma_{size=1..k-1}\, (size*blockCount(size)) = cumKA(k-1) + (k-1)*\, blockCount(k-1)$

Cumulative SLD-histogram. Similar to the cumulative KA histogram we can define a cumulative SLD histogram for the case of simple l-diversity. The cumSLD explains the need for the sumTupleCount measurement in the simple SLD histogram, as it is exactly this value that is summed in order to obtain an exact measurement of how many tuples need to be suppressed when a specific request for a value of *l* is issued. Specifically, for every possible value of *l* (i.e., of distinct number of sensitive values within a group), the cumSLD histogram measures the total number of tuples belonging to groups with a smaller value of *distinctSCount* than l.

$$cumSLD(l) = \Sigma_{dsc=1..l-1}\, (sumTupleCount(dsc))$$

Usage. To see how the histograms facilitate the task of determining appropriate anonymization schemes, take for example the first 10 pairs of the KA and the cumKA histogram from the Adult data set with quasi identifier set {*Age*, *Work class*, *Race*} and generalization scheme A.L1, W.L1, R.L0 depicted in Figure 2.8.

size	KA-histogram	cumKA histogram
1	26	0
2	16	26
3	10	58
4	5	88
5	8	108
6	6	148
7	4	184
9	5	212
10	4	257
11	1	297

Figure 2.8 The 10 first pairs for the KA and cumKA histogram over the Adult data set with quasi identifier set {Age, Work class, Race} and generalization scheme A.L1, W.L1, R.L0.

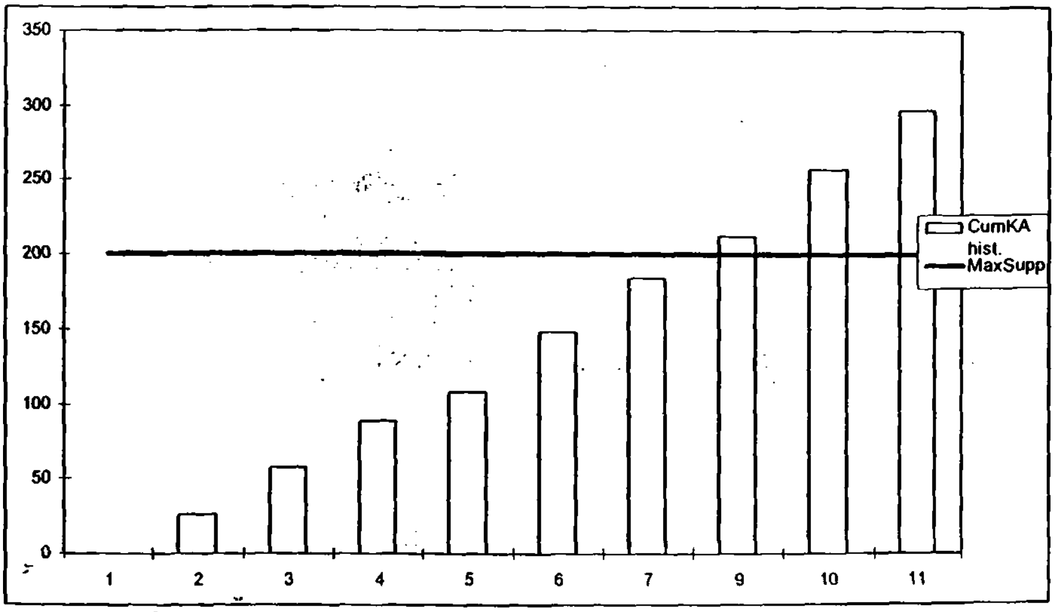


Figure 2.9 CKAb. The 10 first pairs for cumKA histogram depicted as graph along with a *MaxSupp* threshold of 200 tuples.

Observe the graphical representation of Fig. 2.9. The depicted histogram is the one of Figure 2.8. Along with it, suppression maximum threshold of 200 is also depicted in the figure. Then, the figure tells us that if we want an anonymization setting where no more than 200 tuples are suppressed, we cannot use a value of k higher than 7. If we want to use a value of k = 8, 9, 10, etc, then we must suppress at least 212, 212, 257, etc tuples, thus violating the constraint on our *MaxSupp*. Therefore, it is evident, that given a fixed generalization scheme and maximum tolerable number of suppressed tuples, we cannot achieve any value of k that we want; on the contrary, there is an upper bound to the anonymization that we can perform, as expressed by the value of k.

Discussion. To compute the size of the lattice with histograms (in bytes), one has to multiply the lattice size |L| with the average size of the histogram per node.

CHAPTER 3. VALIDITY OF THE PROBLEM: EARLY FINDINGS

Is suppression really a problem for the well intended end users? What is the interrelationship between suppression, generalization and anonymity parameters?

So far, related research in the area of generalization has mainly followed a suppression-agnostic approach. Apart from few early papers [Sama01, Swee02a, Bayardo05] that deal with suppression issues, subsequent research was primarily targeted to local or multidimensional recoding techniques where suppression is not an issue. Despite the obvious benefits of these approaches, it is quite possible that the well-meaning end-users cannot utilize the ad-hoc generalizations of the quasi identifier data to perform their data analysis operations and might demand the presentation of data in generalization hierarchies that have been constructed in advance, taking into account the mappings of values that are intuitive to the users. In this case, we lose one of the good properties of multidimensional and local recoding which is the fitting of outliers in convenient areas. The presence of outliers demands either high generalization abstractions or suppression.

The goal of this section is to fill the aforementioned gap and assess how suppression, generalization and anonymity criteria are related. The main desideratum is the answer to the questions: "how high should we go in the hierarchies to achieve low suppression?", or, "how is the anonymity criterion (e.g., k in k-anonymity) affecting the percentage of suppressed tuples?", or "assuming that we have a strict anonymity criterion (e.g., a high value of k)t and significant ascending in the hierarchy, what percentage of the data set is eventually suppressed?". The answers to these questions are important, since (a) they reveal some knowledge that the current body of knowledge has not addressed and (b) they can guide us through the subsequent negotiation process towards acceptable solutions.

To assess how suppression, generalization and anonymity criteria are related, we start with a simple, but illustrative test. We chose the simplest anonymity criterion, k-anonymity, as our privacy criterion. The criterion of k-anonymity has a simple test: it requires that every group formed by a combination of values by the quasi-identifiers contains at least k tuples. So, if we want to measure the extent of suppression in a data set, for a given generalization scheme, we need to measure the tuples that fall in groups with size smaller than k. Again, this is the simplest test that can be performed for generalization techniques; out of the more elaborate tests (like l-diversity, t-closeness or other) that require extra constraints on the statistical properties of the sensitive values of each group, we also work with l-diversity, too. L-diversity comes in several flavours of increasing complexity; its simplest variants require that every sensitive value in a group is repeated no less time than a certain percentage; or else, that there are at least *l* distinct values in the group.

For our experiments, we work with (a) the Adult data set [UCI] and (b) the PUMS data set [IPUMS].

The goal of the experiments was to measure the number of suppressed tuples as we increase (a) the generalization height and (b) the size of the quasi identifier.

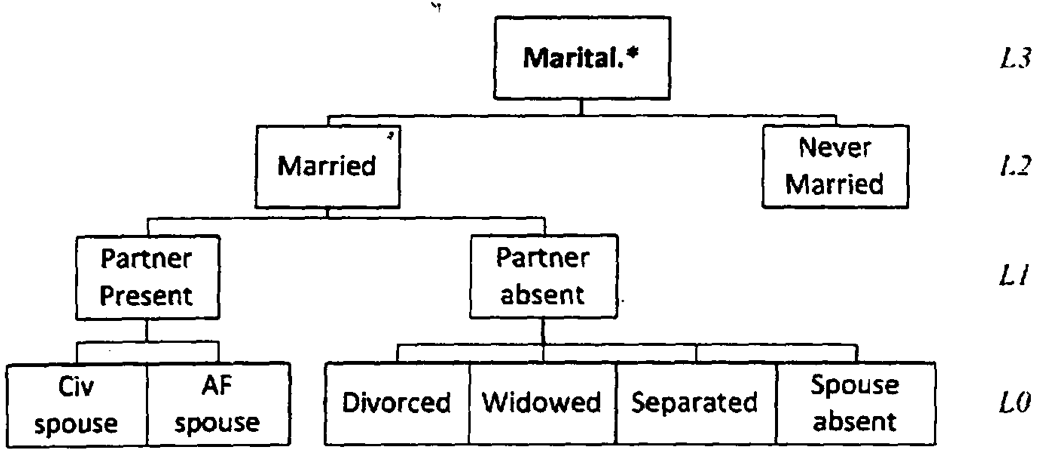


Figure 3.1 The hierarchy for the QI dimension *Marital Status*

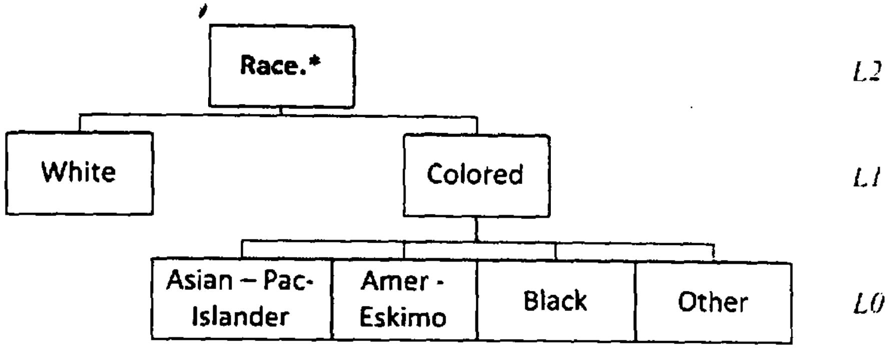


Figure 3.2 The hierarchy for the QI dimension Race

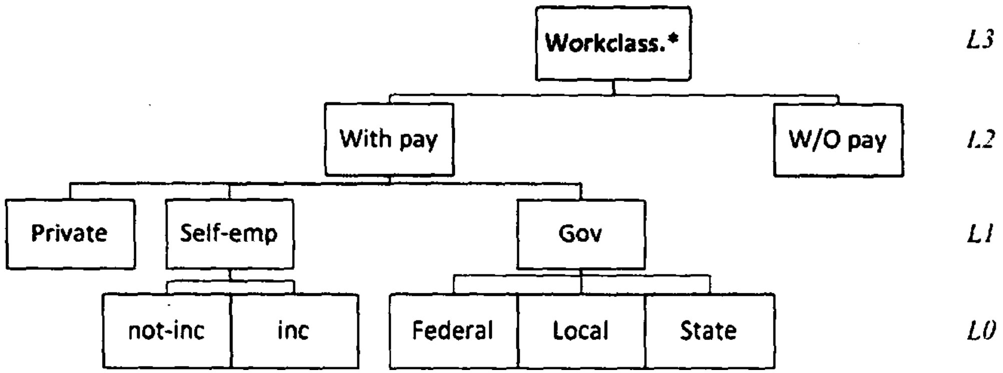


Figure 3.3 the hierarchy for the QI dimension Work class

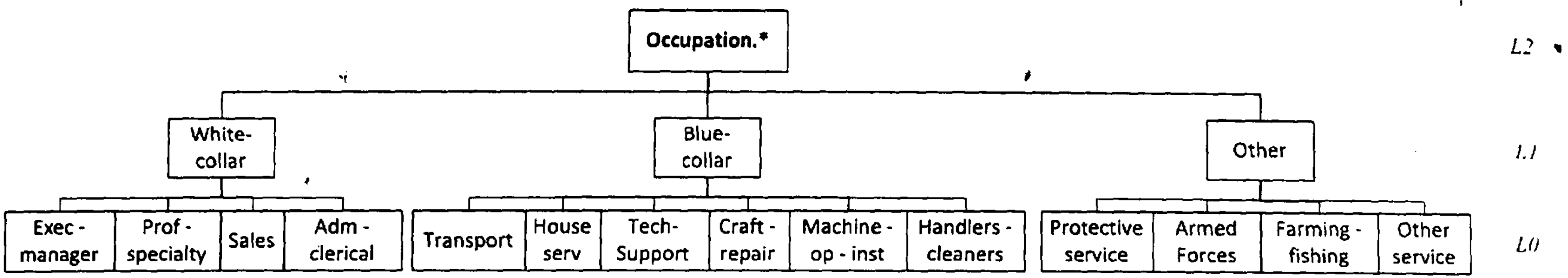


Figure 3.4 The hierarchy for the QI dimension *Occupations*

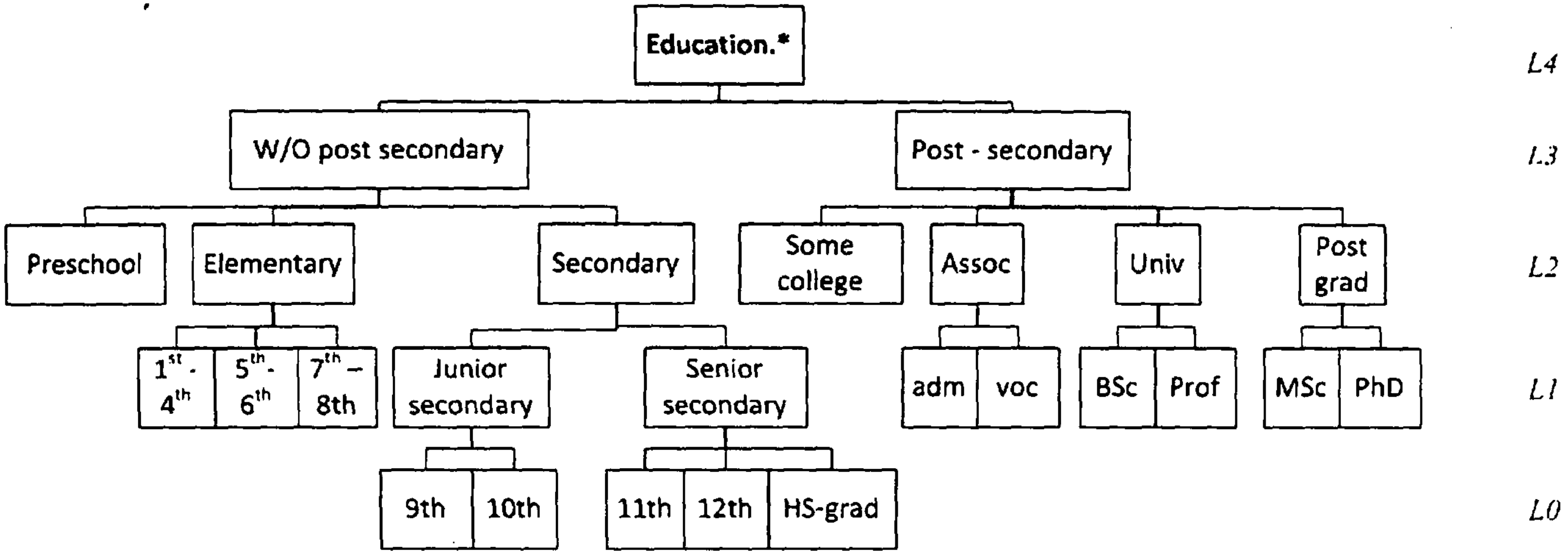


Figure 3.5 The hierarchy for the QI dimension *Education*

3.1. Working with the Adult data set

The first data set that we consider is the Adult data set [UCI] (a.k.a census income dataset), which the most common data set used in the related literature. The dataset in its cleansed version (after uncertain and NULL values are removed) comprises 30162 tuples of the 1994 USA census. Since we require levels of generalization for the quasi-identifiers, we assigned hierarchies determined in advance to the quasi identifiers of the data set. We reused the hierarchies of [FuWY05] which we found reasonable. The hierarchies for the fields *Education*, *Occupation*, *Marital status*, *Work class*, and *Race* are depicted in Figure 3.1-Figure 3.5. Attribute *Age* is organized in years, 5-year intervals, 10-years intervals, 20-year intervals and *. We have used the attribute *Hours per Week* as the sensitive attribute. Attributes *Gender* and *Salary* were not used due to their very small domain of values (Salary has only two values, higher or lower than 50K). Attribute *Native Country* is also not used, since out of the 30162 tuples of the Adult data set, the 27625 tuples have a value of USA, which practically means that the attribute is pretty much like being at level *all*.

An interesting experimental parameter was the choice of attributes for each quasi-identifier size. Since we need to experiment with different sizes of the quasi-identifier set of attributes, we needed to test the attributes on their *grouping power*: ***If an attribute tends to drive an anonymization scheme with large equivalence classes, this means that the possibilities for suppression are smaller than with the case of an attribute that drives the anonymization towards groups with small equivalence classes***. So, we have sorted the attributes according to their grouping power via the following procedure.

- For every attribute, we fix all other attributes at level all and keep this attribute at the most detailed level.
- Then, for every value of this attribute, we count the number of tuples that have this value and group the results per group size. For example, Table 3.1 lists attribute *Marital Status* at the most detailed level, as well as the first 10 rows for attribute *Age* that gave the following histograms:

  Clearly, attribute Age drives the anonymization towards many small-sized equivalence classes compared to attribute Marital Status. Practically, this is due to the fact that the domain of attribute Age is much larger, thus resulting in many small groups. We considered the smallest of these values (which gives us the smaller group that can be formed) as our discriminatory criterion. This is an approximate estimation for the grouping power of the attribute. We avoided the average value of the first few results, since this can be misleading (as for example, in the aforementioned case of attribute *Native Country*).

- Then, we sort the attributes with respected to the size of smallest group in ascending order.

Table 3.1 Histograms for attributes *Marital Status* and *Age*.

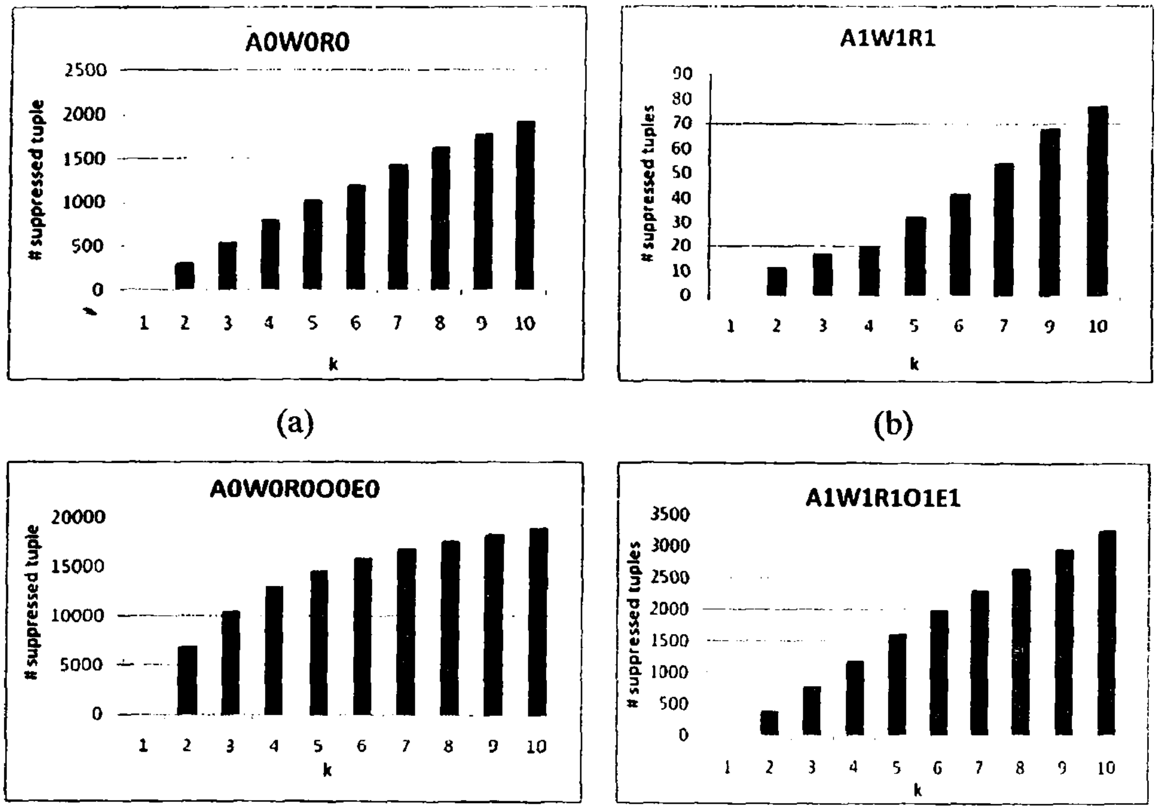
Marital_status level0, size of group	Number of groups with this size
21	1
370	1
827	1
939	1
4214	1
9726	1
14065	1

Age level0, size of group	Number of groups with this size
1	1
3	2
5	1
7	1
8	1
13	1
14	1
15	1
16	1
20	1

The resulting order of attributes was: *Age* with a smallest group size of 1, *Occupation* with a smallest group size of 9, *Work Class* with a smallest group size of 14, *Marital status* with a smallest group size of 21, *Education* with a smallest group size of 45, and, finally, *Race* with a smallest group size of 231. We decided to mix attributes with high and low grouping power as much as possible in our experiments, thus resulting in the final order of attributes which is *Age*, *Work class*, *Race*, *Occupation*, *Education*, *Marital status*, *Native Country*. So, for example, when we say that the quasi-identifier size is 3, the quasi-identifier attributes are *Age*, *Work class*, *Race*, when we say that the quasi-identifier size is 6, the quasi-identifier attributes are *Age*, *Work class*, *Race*, *Occupation*, *Education*, *Marital status*.

3.2. K-anonymity for the Adult data set

In this subsection, we report on our findings for the relationship of maximum allowed suppression, privacy preservation (expressed by the k-anonymity criterion) and level of generalization over the Adult data set. In our experiments, we measure the number of suppressed tuples per node of the lattice of the Adult set. We have conducted this experiment for all the possible values of QI between 2 and 7. We discuss the cases of QI = 3 and 5 that are the most characteristic – the rest of the case behave similarly to the observations we make here.



(a) (b)

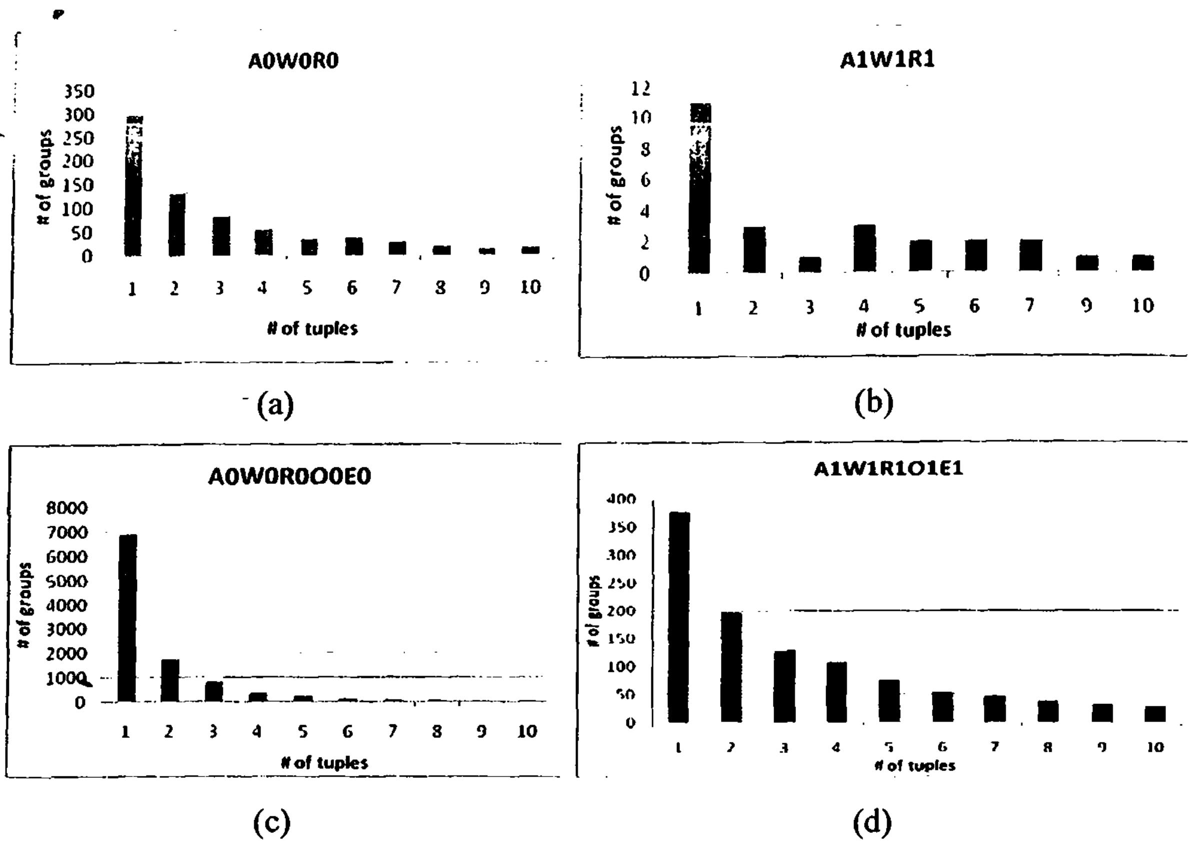
(c) (d)

Figure 3.6 Cumulative histograms for different levels of generalization for |QI| size of 3 (a,b) and 5(c,d).

Figures 3.6a,b depict the cumKA histograms – i.e., the number of tuples to be suppressed per value of k for two different levels of generalization. The size of the QI is 3 and comprises the attributes *Age, Work Class*, and *Race* (in this order). In Figure 3.7a we depict the histogram for the case where no generalization takes place (denoted as A0W0R0) and Figure 3.7b depicts the histogram for the case where all attributes are generalized by one level (denoted as A1W1R1). We observe that (a) there is a practically linear increment of suppressed tuples per value of k (i.e., the suppression increases rather slowly with k) and (b) once we generalize all the dimensions by one level, the suppression is reduced by 2 orders of magnitude. In

Figures Figure 3.6c,d we can see the cumKA histograms for size of QI equal to 5; specifically, the attributes considered are *Age, Work class, Race, Occupation and Education*. It is worth noting that the increase of the QI size by 2 dramatically increases the amount of suppression by one order of magnitude. Interestingly, on the case where QI=5 and no generalization takes place, the amount of suppression surpasses 50% of the data set for a value of k=6. The vase where all dimensions are generalized by one level presents a more linear increase of the suppression with the increase of k and demonstrates amounts of suppression lower by one order of magnitude than the case of no suppression.

Figures 3.7a,b depict the KA histograms – i.e., the number of groups per group size for two different levels of generalization. In Figure 3.7a we depict the histogram for the case where no generalization takes place (denoted as A0W0R0) and Figure 3.7b depicts the histogram for the case where all attributes are generalized by one level (denoted as A1W1R1). We observe that there is an exponential reduction in the number of groups per group size within each histogram. Most importantly, however, if one compares the two generalization levels, there is a reduction by a scale factor of 30 for the number of groups of the same size between the two generalization schemes! The same applies for the cumulative behavior of the histogram too. For example, if we want to achieve 3-anonymity, we have to suppress 554 tuples (1*296+2*129) for the case of A0W0R0 and 17 tuples (1*11+2*3) for the case of A1W1R.



(a) (b)

(c) (d)

Figure 3.7 Number of groups per group size for different levels of generalization for |QI| size of 3 (a,b) and 5(c,d).

In Figure 3.7c,d we can see the KA histograms for size of QI equal to 5; specifically, the attributes considered are *Age, Work class, Race, Occupation and Education.* One can observe the following:

- The exponential decrease of number of groups as the size of group increases is retained
- This phenomenon applies to both cases of no generalization and generalization by one level
- Most importantly, one can observe a significant increase in the number of suppressed tuples between the cases of (a)-(b) with |QI|=3 and (c)-(d) with |QI|=5. For example, achieving 3-anonymity in the latter case requires suppressing 10458 tuples (1*6920+2*1769) for A0W0R0O0E0 and 1619 tuples for (1*887+2*366) level A1 W1R1O1E1.

In Figure 3.9 we can see the full lattice for the case of QI=3 (*Age*, *Work class*, *Race*). The numbers that annotate each node show the number of suppressed tuples introduced by the node's generalization scheme for 3-anonymity. Figure 3.10 depicts the respective information for QI = 5 (*Age, Work class, Race, Occupation and Education*) for a subset of the full lattice, and specifically, for the lattice between the generalization schemes 00000 and 11111.

The case of the partial lattice. Before proceeding, we would like to justify the introduction of the partial lattice as one of the means of our experimental method. One of the problems we have faced when comparing findings for different sizes of the quasi identifier set is that the results are not directly comparable. This is due to two main reasons: (a) the size of the lattice differs significantly and (b) the reported numbers of suppressed tuples also differ significantly due to the fact that as the QI size grows, the number of groups formed grows too, and each group shrinks in size as a result (thus, for a fixed k, the number of suppressed tuples grows as the |QI| increases). Although this is a clear and well expected result, we would like to be able to compare the two cases to the extent that this is possible. We observed that if we would focus on the sublattice between 00...0 and 11...1, we had a lattice of comparable size to the lattice of QI = 3 and a quite good approximation of the behavior of the suppression process for the full lattice. In Fig. 3.8 we depict the average number of suppressed tuples per level for the full and partial lattice; as one can see the difference is significant only for the case of H5 (where the partial lattice has only one node).

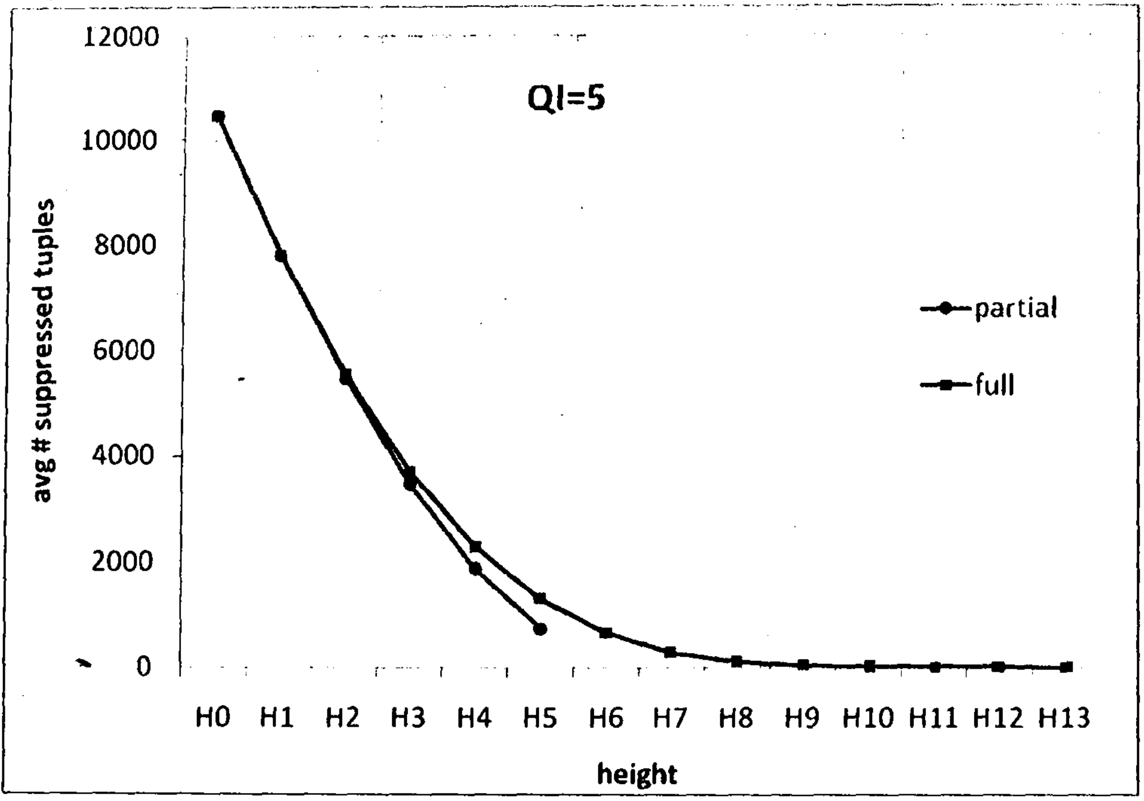


Figure 3.8 Average number of suppressed tuples over different heights for 3-anonymity and QI size of 5 for (a) the full lattice and (b) the partial lattice of the data set.

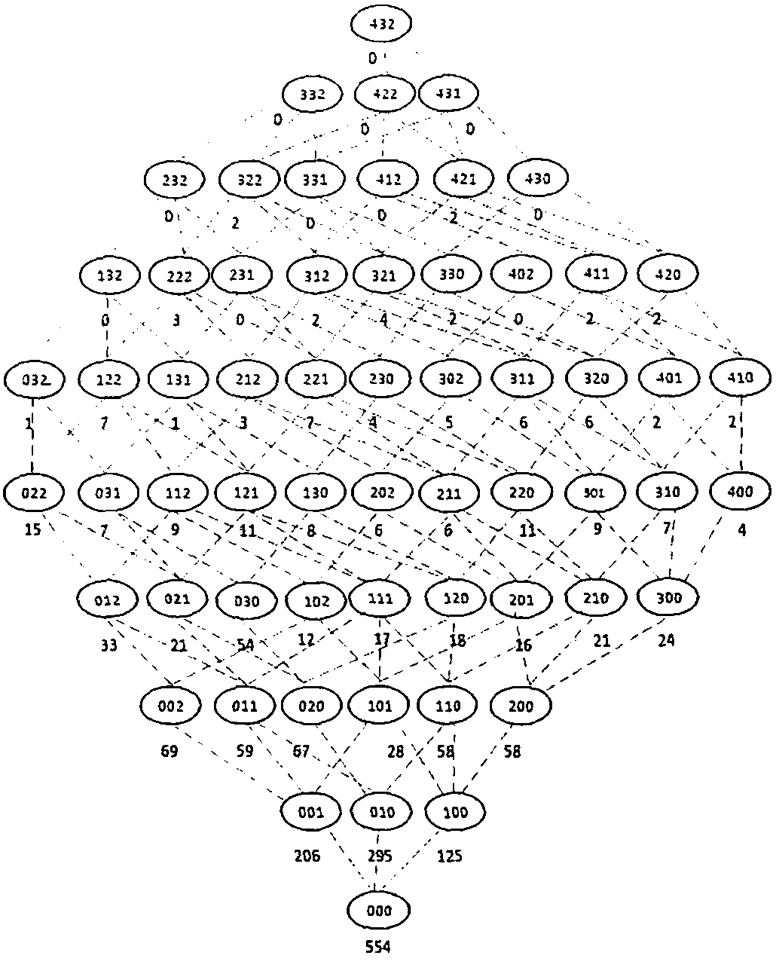


Figure 3.9 Full lattice with suppressed tuples for quasi identifier set of size 3. The QI is *Age*, *Work class*, *Race*.

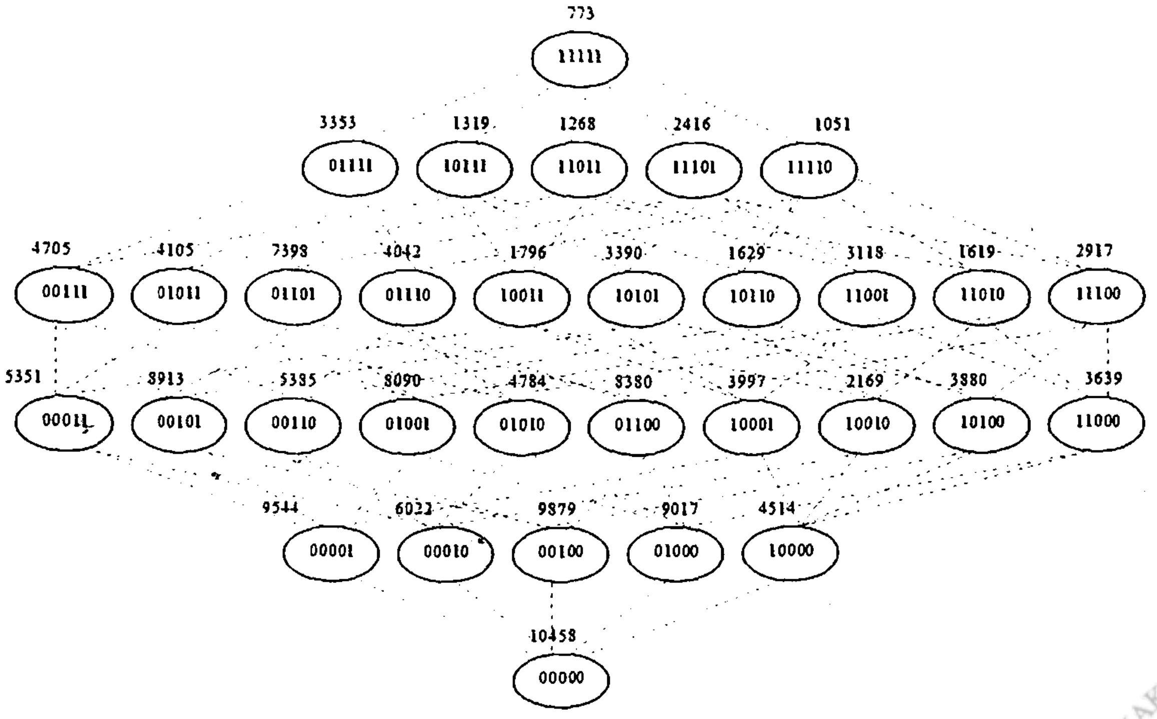


Figure 3.10 Sub-Lattice (between 00000 and 11111) with suppressed tuples for quasi identifier set of size 5. The QI is *Age, Work Class, Race, Occupation, Education*.

3.2.1. Comparison of different levels of generalization with fixed k and QI size

Observe Table 3.2 that compares the two lattices level by level. For each layer of nodes we list the minimum, average and maximum numbers of nodes suppressed, the fractional decrease over the non-generalized data set and the decrease in the suppression with respect to the previous layer. Figure 3.11 depicts the average number of suppressed tuples per level graphically.

Table 3.2 Number of suppressed values for 3-anonymity as (a) the height of the generalization increases and (b) the size of the quasi identifier set increases (for |QI| = 3 and 5)

	|QI| = 3					|QI| = 5				
	Min	**Avg**	**Max**	**Avg % over full**	**% over previous**	**Min**	**Avg**	**Max**	**Avg % over full**	**% over previous**
H0	554	554	554	1,83	-	10458	10458	10458	34,67	-
H1	125	207	295	0,69	62,33	4514	7795	9879	25,84	34,15
H2	28	56	69	0,19	72,92	2169	5459	8913	18,10	42,80
H3	12	24	54	0,08	57,52	1619	3472	7398	11,51	57,22
H4	4	8	15	0,03	64,79	1051	1881	3353	6,24	84,53
H5	1	4	7	0,01	52,66	773	733	733	2,43	156,67
H6	0	2	4	0,01	58,50	-	-	-	-	-

For each row of Table 3.2, we denote with '**avg % over full**' the fraction of the average number of suppressed tuples of the specific height (listed in column 'Avg') over the number of tuples of the whole data set. This measure allows us to see the gradual degradation of the number of suppressed tuples as we ascend the lattice.

Also, we denote with '**% over previous**' the fraction:

$$\frac{|\text{suppressed tuples of previous level}| - |\text{suppressed tuples of current level}|}{|\text{suppressed tuples of previous level}|}$$

This measure allows us to see the gain from ascending one level up in the lattice each time.

The study of Table 3.2 presents the following observations:

- *Comparison of different levels for the same QI.* It is clear that as the height increases the number of suppressed tuples drops with a high rate (observe also

Figure 3.11 where this is graphically depicted). Clearly, there is a point after which the climbing of the lattice is not further required; this practically happens little after the middle of the lattice's height (at height 4 for the 7 levels of QI=3 and height 8 for the 14 levels of QI = 5 – see Figure 3.8 for the latter.

- *Comparison of different QI sizes.* If one compares the average and the minimum numbers of suppressed tuples per height, one can see that the case of QI=5 presents numbers that are between 18 and 262 times(!) higher for levels H0 to H4. For this range of levels, the higher the level, the higher the suppression for QI=5. Remember that Figure 3.8 also depicts the results for the full lattice; with the exception of level H5 which has an average number of suppressed tuples twice the size of the partial lattice, the observations are practically similar.
- *Not all attributes are born equal.* Finally, observe that the range of values between minimum, maximum and average suppression per level is quite wide. Interestingly, in the low levels of generalization (which are much more interesting, because this is where we really want our solutions to be found), the careful choice of generalization scheme can yield approximately half the suppressed tuples than the average case. As the height increases, the importance of this choice remains significant albeit of less importance. The fact that the generalization of some attributes leads to a higher reduction of the number of suppressed tuples is due to the fact that a generalization over an attribute with a large domain reduces main small groups at the detailed level to coarser groups at the generalized level, producing, thus, higher opportunities for the reduction of suppression.
    - Observe, for example, level H3 for QI=3. Node 102 has the smallest number of suppressed tuples (12). It is interesting to notice its parents at level H2: node 002 suppressed 69 tuples (much more than node 102), whereas node 101 suppressed 28. In other words, the best node is produced by (a) generalizing attribute *Age*, (b) not touching attribute *Work Class* and (c) slightly ascending over attribute *Race*. At the same time, the maximum number of suppressed tuples at level H3 is attained by node 030, which does the exact opposite of node 102: it only generalizes (a lot, at level 3) attribute *Work Class*.
    - At the same time, at level H3 for QI=5 (*Age*, *Work class*, *Race*, *Occupation*, *Education*) we can observe that (again) the nodes with the largest and

smallest number of suppressed tuples are practically complementary: node 01101 yields a suppression of 7398 tuples and node 11010 yields a suppression of 1619 tuples. Observe also what happens when we generalize attribute *Work Class*: very small reductions to suppression are produced in almost all occasions when we move from a node without generalization of *Work class* to a node that generalizes *Occupation* (except in the case of the combination of *Occupation* with *WorkClass*) for practically all the levels. This is mainly due to the fact that moving from L0 to L1 for *Work Class* (a) does not involve the values: Private, (b) has a rather small grouping for the values under Self-Employed and, thus, (c) ultimately reduces to grouping the government jobs under value 'Gov' at L1.

Also, observe the behavior of attribute *Occupation* (4th in the numbering of attributes). Apparently, it pays off to ascend from H0 to H1, but not really to ascend from H1 to H2, unless in combination with attribute *Age*. At the same time, ascending from the nodes of H2 with no generalization for *Occupation* to H3 at nodes that do generalize *Occupation* practically reduce suppression in half(!). In other words, it appears that Age is the dominant attribute to consider for suppression reduction and that *Occupation* demonstrates different behavior at different levels, depending on the rest of the generalized attributes.

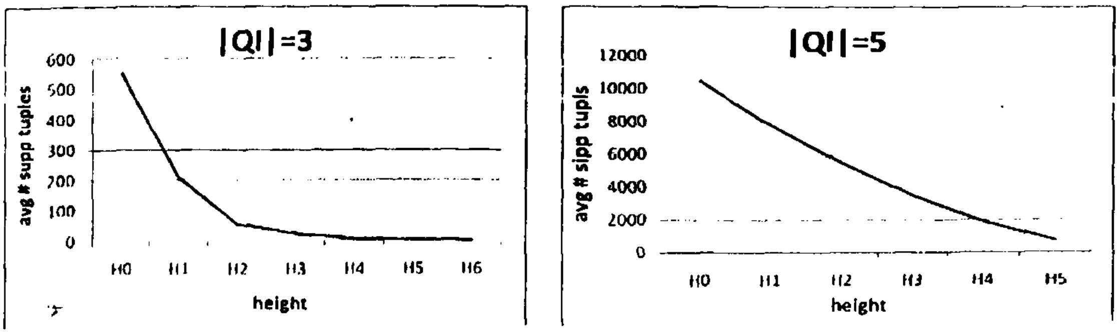


Figure 3.11 Average number of suppressed tuples over different heights for 3-anonymity. The QI size of 3 refers'to the full lattice and the QI size of 5 to the partial lattice of Figure 3.2.4.

3.2.2. Comparison of different values of k and height with a fixed QI size

In this subsection, we report on our findings when comparing different values of k for the privacy criterion of k-anonymity for their effect on the number of suppressed tuples. For each layer of nodes we list the minimum, average and maximum numbers of suppressed tuples, for a height up to H6, for QI sizes of 3 (Table 3.3) and 5 (Table 3.4). The results are also graphically depicted in Figure 3.12 and Figure 3.13.

Table 3.3 Minimum, maximum and average number of suppressed tuples for k=3,10,25 and QI size of 3 over the full lattice.

|QI|=3 (lattice up to height H6)									
	k=3			k=10			k=25		
	Min	**avg**	**max**	**min**	**avg**	**max**	**min**	**avg**	**max**
H0	554	554	554	1921	1921	1921	4578	4578	4578
H1	125	209	295	522	1030	1357	1184	2546	3573
H2	28	57	69	170	352	508	610	1153	1926
H3	12	24	54	51	148	484	195	419	1236
H4	4	8	15	28	45	94	56	127	222
H5	1	4	7	2	19	37	14	48	105
H6	0	2	4	0	9	23	14	21	40

Table 3.4 Minimum, maximum and average number of suppressed tuples for k=3,10,25 and QI size of 5 over the partial lattice.

|QI|=5 (PARTIAL lattice)									
	k=3			k=10			k=25		
	Min	**avg**	**max**	**min**	**avg**	**max**	**min**	**avg**	**max**
H0	10458	10458	10458	18916	18916	18916	25945	25945	25945
H1	4514	7795	9879	10944	15974	18801	16492	22282	25945
H2	2169	5459	8913	6151	12684	18325	10655	18624	25945
H3	1619	3472	7398	4824	9291	17359	8516	14867	25084
H4	1051	1881	3353	3990	6049	10141	7520	10923	16818
H5	773	733	733	3259	3259	3259	6712	6712	6712

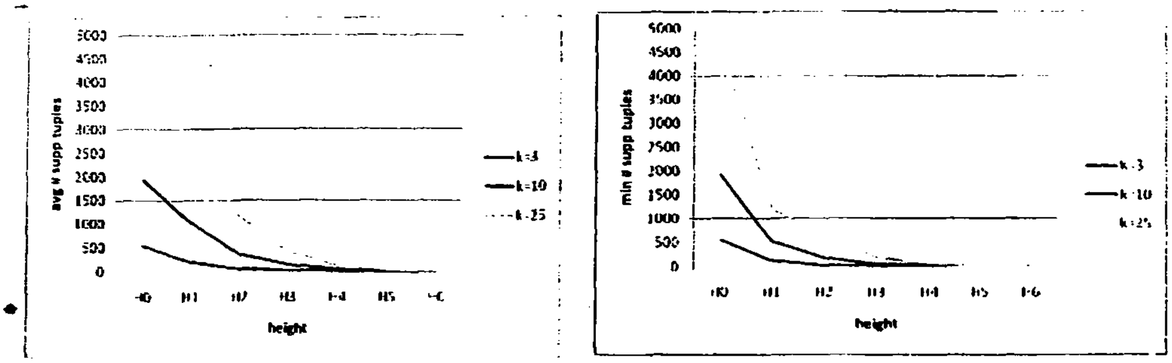


Figure 3.12 Average and minimum number of suppressed tuples over different heights for a QI size of 3 and different k for k-anonymity. The reported numbers refer to the full lattice.

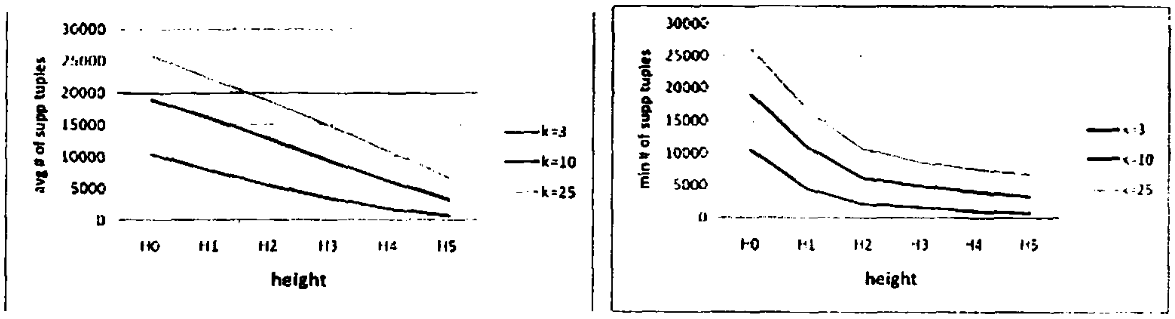


Figure 3.13 Average and minimum number of suppressed tuples over different heights for a QI size of 5 and different k for k-anonymity. The reported numbers refer to the partial lattice.

Our observations can be summarized as follows:

- *The effect of k to the suppression.* By comparing the same lines of the two tables over different values of k, one can clearly see that the effect of the privacy criterion (here: k for k-anonymity) to the amount of suppressed tuples is practically analogous to the amount of suppression performed.

  As the height is small and the number of suppressed tuples significant (in fact, higher than the value of k, i.e., till height H3 included), the ratio of the minimum number of suppressed tuples between k=3 and k=10, as well as k=10 and k=25 increases slowly (we chose the minimum since it is the value of the best solution) and remains close to the fraction of the two k's.

Table 3.5 Ratio of minimum values for different values of k, QI size and height

	QI = 3		|QI|=4		QI = 5		|QI|=6	
	min(k=10) / min (k=3)	min(k=25) / min(k=10)	min(k=10) / min (k=3)	min(k=25) / min(k=10)	min(k=10) / min (k=3)	min(k=25) / min(k=10)	min(k=10) / min (k=3)	min(k=25) / min(k=10)
H0	3,47	2,38	2,86	1,56	1,81	1,37	1,57	1,18
H1	4,18	2,27	3,14	2,02	2,42	1,51	1,91	1,34
H2	6,07	3,59	3,97	2,28	2,84	1,73	2,17	1,47
H3	4,25	3,82	4,75	2,27	2,98	1,77	2,49	1,64
H4	7	2	6,07	3,59	3,8	1,88	2,61	1,68
H5	2	7	4,25	3,12	4,22	2,06	3,03	1,71

- *The effect of height increase over the number of suppressed tuples is the same for different k's.* Observe Figure 3.12 and Figure 3.13. All the lines are practically parallel; in other words, independently of k, the trend of suppression and the height increases is the same. Observe also, that when minimum values are concerned, the changes are slightly steeper than in the case of average values; however this observation is of secondary importance.
- *Computing the fraction between minimum and average number of suppressed tuples.* Concerning QI=3, the fraction of the average number of suppressed tuples over the minimum number of suppressed tuples is approximately around 2 – and, in a couple of cases it raises up to 3 times. When we move to QI=5, the respective fraction ranges on average between 1.8 to 1.5 – dropping as k increases. In other words, it is still important to carefully pick a good solution with a price of 50% – 100% with respect to the average cost. Still, as QI and k increase, the importance of this decision diminishes.

For completeness, we also list the average and the min numbers of suppressed tuples for the full lattice of QI=5 in Table 3.6, Figure 3.14 and Figure 3.15.

Table 3.6 Average number of suppressed tuples for k=3,10,25 and QI size of 5 over the full lattice

	|QI|=5 (FULL lattice). avg and min #suppressed tuples per level					
	k=3		**k=10**		**k=25**	
	avg	**min**	**avg**	**min**	**avg**	**min**
H0	10458.00	10458	18916.00	18916	25945.00	25945
H1	7795.20	4514	15973.80	10944	22282.00	16492
H2	5537.07	2169	12734.20	6151	18954.20	10655
H3	3711.88	1123	9652.73	3468	15463.70	6599
H4	2296.34	716	6804.24	2065	11929.88	4247
H5	1295.15	322	4400.43	1160	8539.00	2471
H6	644.26	108	2551.32	578	5524.31	1257
H7	282.98	41	1288.55	230	3173.08	648
H8	110.42	8	554.97	60	1535.83	263
H9	40.52	2	212.72	14	631.24	26
H10	14.16	0	72.94	0	223.06	12
H11	4.92	0	25.14	0	78.46	0
H12	1.42	0	10.09	0	26.30	0
H13	0.27	0	2.87	0	6.27	0
H14	0.00	0	0.00	0	0.00	0
H15	0.00	0	0.00	0	0.00	0

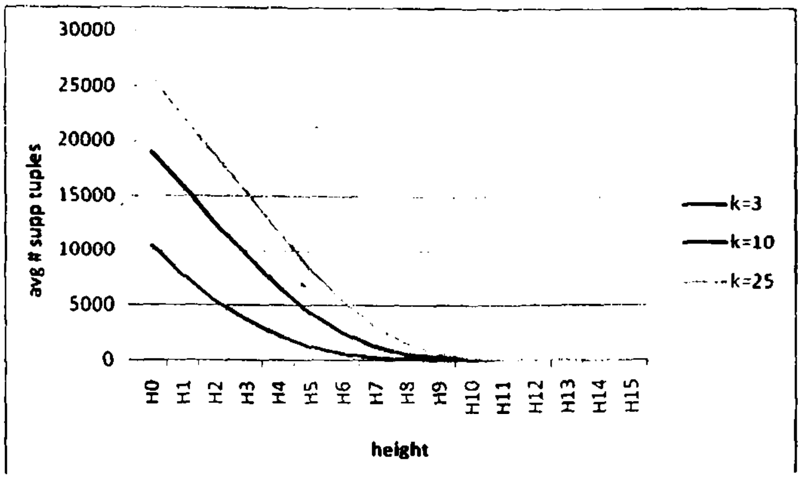


Figure 3.14 Average number of suppressed tuples over different heights for a QI size of 5 and different k for k-anonymity. The reported numbers refer to the full lattice.

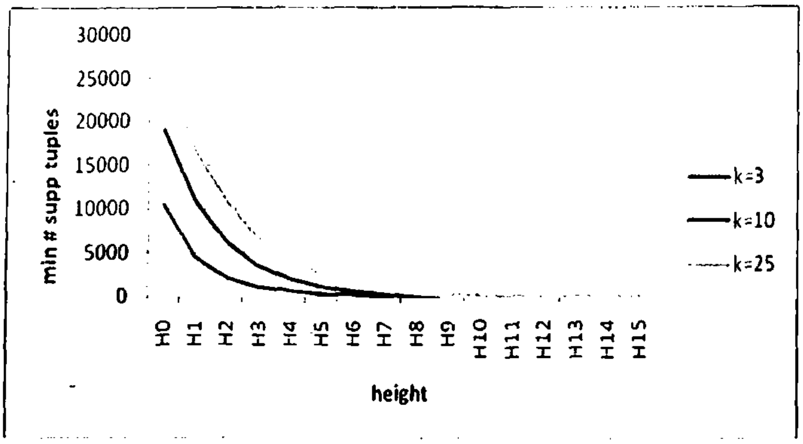


Figure 3.15 Min number of suppressed tuples over different heights for a QI size of 5 and different k for k-anonymity. The reported numbers refer to the full lattice.

3.2.3. Comparison of different QI sizes (over different levels) with a fixed value of k

In this subsection, we focus our observations in the effect of increasing the QI size over the amount of suppressed tuples. We fix the level of k-anonymity to k = 3 and present our results per different levels of generalization and QI size.

Our observations can be summarized as follows:

- Clearly, different QI sizes at the same level have on average an increase of the scale of 2 -3 times, for large volumes of suppressed tuples. This scale factor changes as the volume of suppressed tuples drops
- Moreover, it is clear that statistically tolerable amounts of suppressed tuples are attained slower as the size of QI grows. For example, the suppression percentage falls under 1% of the total volume of data at height H1 for QI = 3, H3 for QI = 4, H6 for QI = 5 and after H8 for QI = 6.
- *The most important observation is that a QI of size n drops to the levels of suppression of the QI of size n-1 around 3-4 levels of generalization later for smaller QI's and 1-2 levels for larger QI's.*

3.2.3.1. Partial lattices

Table 3.7 Average number of suppressed tuples and percentage over the full data set for 3-anonymity for different QI sizes over the partial lattice.

	|QI|=3		|QI|=4		|QI|=5		|QI|=6	
	Avg	Avg % over full	Avg	Avg % over full	Avg	Avg % over full	Avg	Avg % over full
H0	554	1,836	3297,0	10,9	10458,0	34,7	15318,0	50,8
H1	208,66	0,691	1847,8	6,1	7795,2	25,8	12808,7	42,5
H2	48,5	0,160	803,3	2,7	5458,8	18,1	10342,5	34,3
H3	17	0,056	217,5	0,7	3471,9	11,5	7958,4	26,4
H4	-	-	47,0	0,2	1881,4	6,2	5740,1	19,0

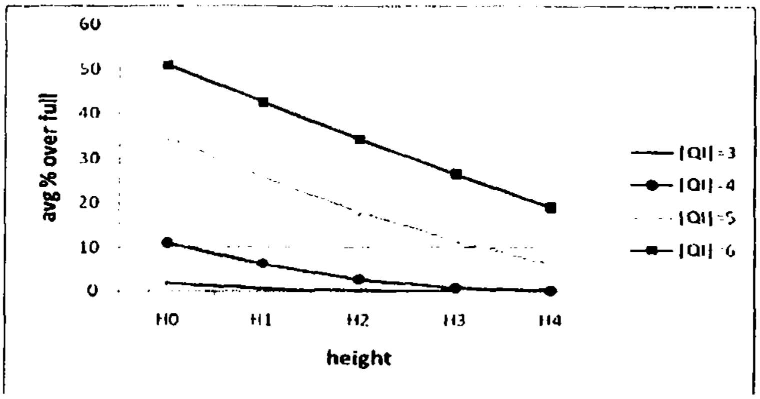


Figure 3.16 Percentage of suppressed tuples over different heights for 3-anonymity. The reported numbers refer to the partial lattices for all QI sizes.

Table 3.8 Min number of suppressed tuples over the full data set for 3-anonymity for different QI sizes over the partial lattice.

	min			
	|QI|=3	|QI|=4	|QI|=5	|QI|=6
H0	554	3297	10458	15318
H1	125	1042	4514	8304
H2	28	318	2169	4901
H3	12	110	1619	4023
H4	4	47	1051	3155

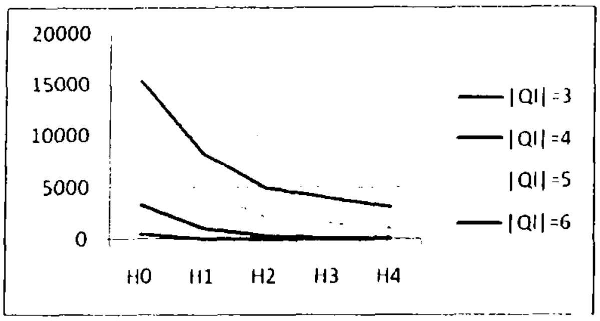


Figure 3.17 Min suppressed tuples over different heights for 3-anonymity. The reported numbers refer to the partial lattices for all QI sizes.

3.2.3.2. Full lattices

Table 3.9 Average number of suppressed tuples and percentage over the full data set for 3-anonymity for different QI sizes over the full lattice.

	|QI|=3		|QI|=4		|QI|=5		|QI|=6	
	Avg	**Avg % over full**	**Avg**	**Avg % over full**	**Avg**	**Avg % over full**	**Avg**	**Avg % over full**
H0	554,0	1,8	3297,0	10,9	10458,0	34,7	15318,0	50,8
H1	208,7	0,7	1847,8	6,1	7795,2	25,8	12808,7	42,5
H2	56,5	0,2	868,6	2,9	5537,1	18,4	10369,3	34,4
H3	24,0	0,1	354,3	1,2	3711,9	12,3	8105,1	26,9
H4	8,5	0,0	121,0	0,4	2296,3	7,6	6036,7	20,0
H5	4,0	0,0	42,9	0,1	1295,1	4,3	4255,2	14,1
H6	1,7	0,0	15,1	0,0	644,3	2,1	2803,0	9,3
H7	0,7	0,0	6,1	0,0	283,0	0,9	1703,8	5,6
H8	0,0	0,0	2,1	0,0	110,4	0,4	941,1	3,1
H9	0,0	0,0	0,4	0,0	40,5	0,1	465,5	1,5

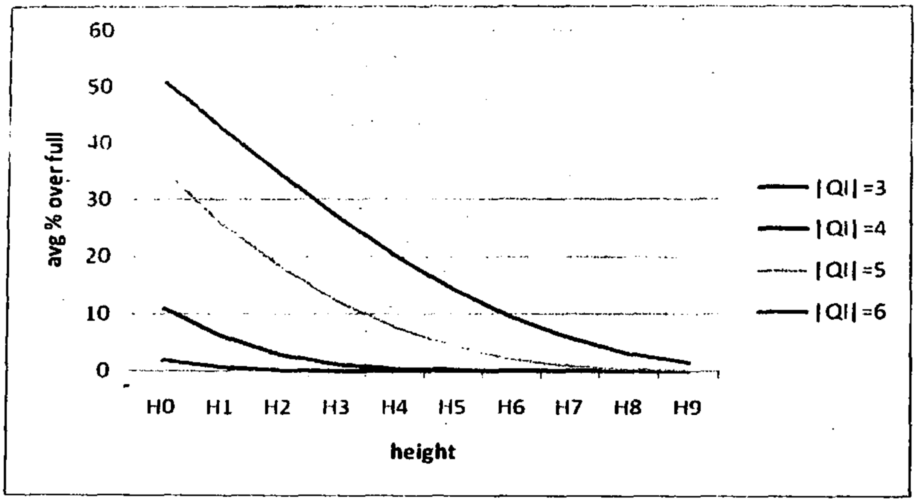


Figure 3.18 Percentage of suppressed tuples over different heights for 3-anonymity. The reported numbers refer to the full lattices for all QI sizes.

Table 3.10 Average number of suppressed tuples and percentage over the full data set for 3-anonymity for different QI sizes over the full lattice.

	Min # of suppressed tuples			
	|QI|=3	|QI|=4	|QI|=5	|QI|=6
H0	554	3297	10458	15318
H1	125	1042	4514	8304
H2	28	318	2169	4901
H3	12	110	1123	2867
H4	4	28	716	1941
H5	1	12	322	1177
H6	0	4	108	629
H7	0	0	41	354
H8	0	0	8	155
H9	0	0	2	33

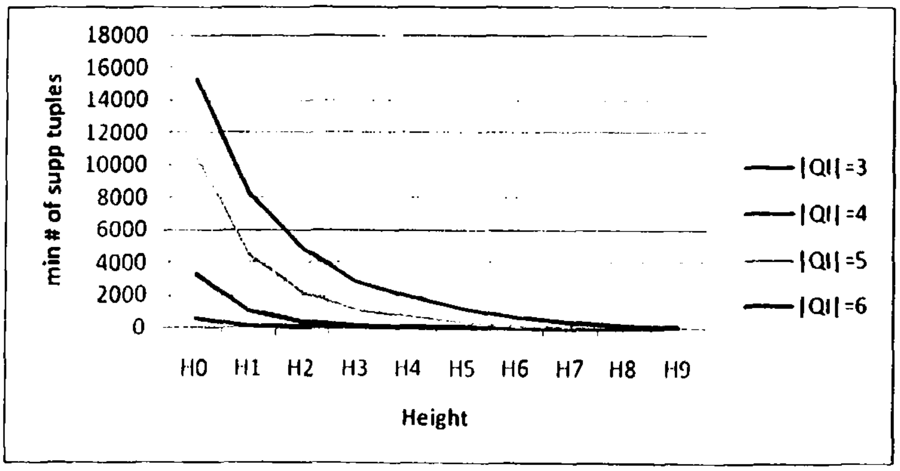


Figure 3.19 Min suppressed tuples over different heights for 3-anonymity. The reported numbers refer to the full lattices for all QI sizes.

3.2.3.3. Selected nodes

Observe also **Table 3.11**, where we compare "homologous" nodes. Since the quasi-identifier size is different, one might possibly argue that the abovementioned comparison is uñfair. So, we compare the following cases:

- *both configurations have a single attribute generalized (QI=3 with nodes 001, 010, 100 vs. QI=5 with nodes 00001, ..., 10000)*: observe how the ranges for QI =3 are all below 1%, whereas the smallest suppression for QI =5 is practically 15%(!)

Table 3.11 Comparison of homologous nodes: (a) absolute numbers and (b) percentage of suppressed tuples over the full data set for 3-anonymity.

	001	010	100	avg	00001	00010	00100	01000	10000	avg
# tuples suppressed	206	295	125	208,66	9879	9017	9544	6022	4514	7795,2
% vs 0	0,68	0,97	0,41	0,69	35,75	29,89	31,64	19,96	14,98	25,84

- *both configurations have three levels generalized (QI = 3 with node 111 vs. QI=5 with nodes 00111, ..., 11100)*: no matter which node of QI=5 we pick, with a generalization of three levels, it is clear that the effect of the size of QI is very important: the best possible suppression of the nodes with QI=5 is 95 times larger than the respective suppression of node 111. At the same time, in order to highlight that almost all of these nodes are important nodes in the lattice, we extend **Table 3.12** with the last column which shows that each of these nodes (with the exception of 00111) provides a significant reduction of the amount of suppressed tuples with respect to the average node of its previous level (i.e., apart from node 00111, all the other nodes would be worthy candidates to consider as generalization schemes if necessary).

Table 3.12 Comparison of homologous nodes: (a) absolute numbers, (b) percentage of suppressed tuples over the full data set for 3-anonymity and (c) improvement over the average of the previous level

	Num. suppressed	% over level 0	% improvement over the avg previous level
111	17	0,056	69,9115
00111	7398	24,527	- 35,519
01011	4042	13,400	25,957
01101	4705	15,599	13,812
01110	4105	13,609	24,803
10011	2917	9,671	46,565
10101	3390	11,239	37,900
10110	3118	10,337	42,883
11001	1629	5,400	70,159
11010	1619	5,367	70,342
11100	1796	5,954	67,100

- QI = 3 with node 111 vs. QI = 5 with node 11111: In Table 3.13, we can see that the top of the "diamond" of the partial lattice, which is the best that the partial lattice can do, is quite low, in both cases -- even despite the fact that a difference of two attributes in the QI size results in a scale factor of 45(!!) for the suppressed tuples. This is an important observation, since one might "safely" restrict the search within the partial lattice for a quick generalization which is not necessarily the optimal.

Table 3.13 Comparison of homologous nodes: (a) absolute numbers and (b) percentage of suppressed tuples over the full data set for 3-anonymity.

	111, QI = 3	11111, QI = 5
# tuples suppressed	17	773
% vs 0	0,056	2,56

3.2.4. Big picture

In the sequel we provide a combined view of all the results of this subsection. We use a diagrammatic technique that combined QI sizes, heights and different values of k and depicts the number of suppressed tuples for every possible combination in a single figure.

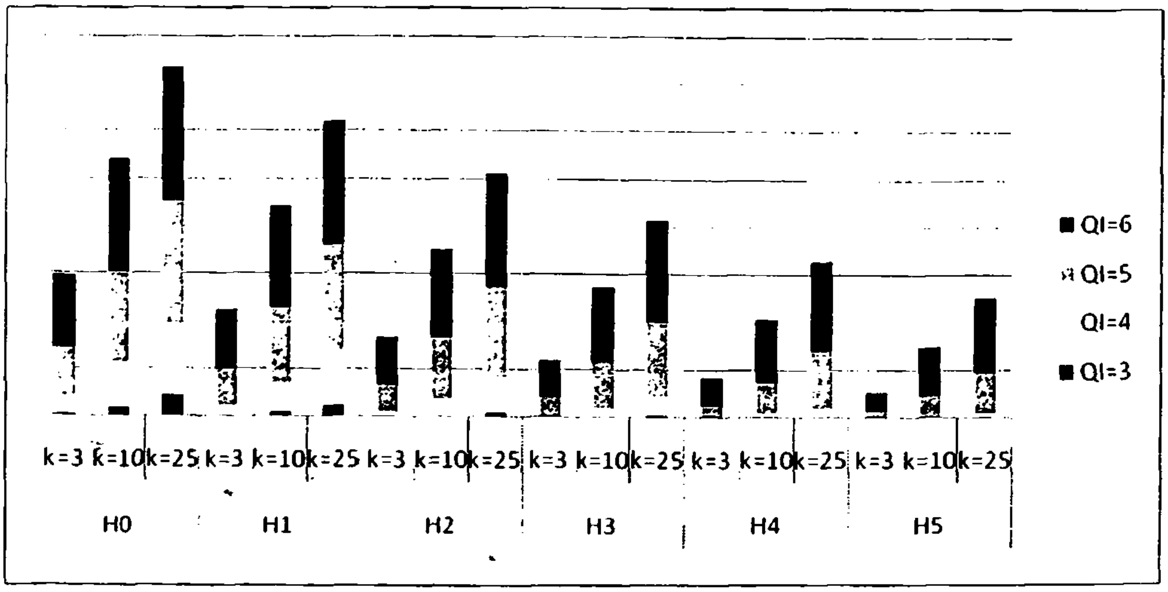


Figure 3.20 Relative volume of suppressed tuples for different combinations of generalization height, k and QI size (each sub-bar depicts the avg number of suppressed tuples *fully* – i.e., not as a differential over the previous sub-bar; thus, it is meaningless to add the different values of sub-bars within a bar). Each vertical interval between horizontal lines corresponds to 10,000 tuples.

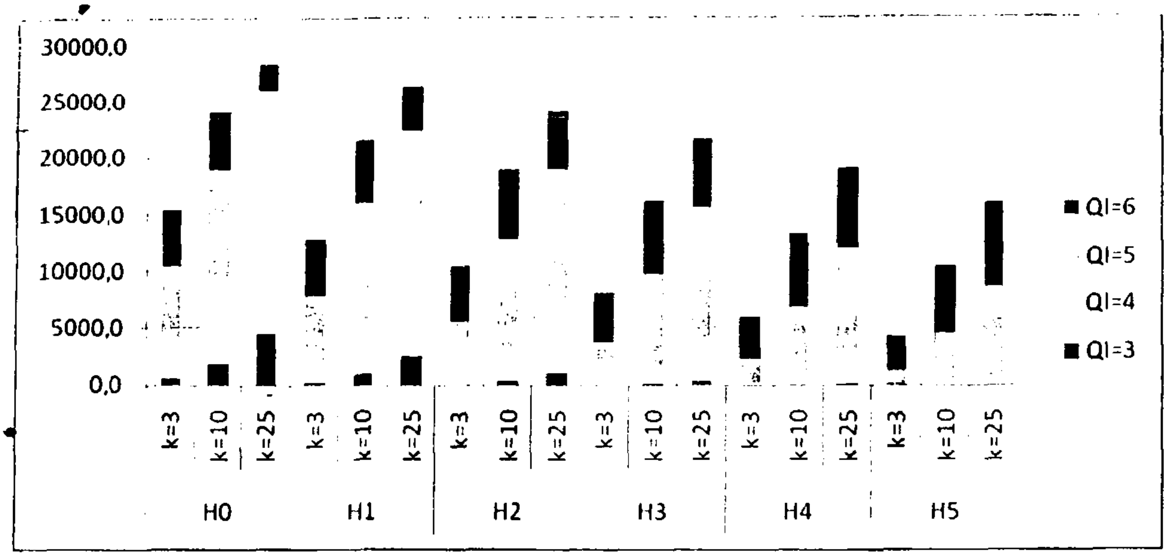


Figure 3.21 Relative volume of suppressed tuples for different combinations of generalization height, k and QI size (each sub-bar depicts the avg number of suppressed tuples *incrementally* – i.e., with each bar as a differential over the previous sub-bar; thus, it makes sense to add the different values of sub-bars within a bar).

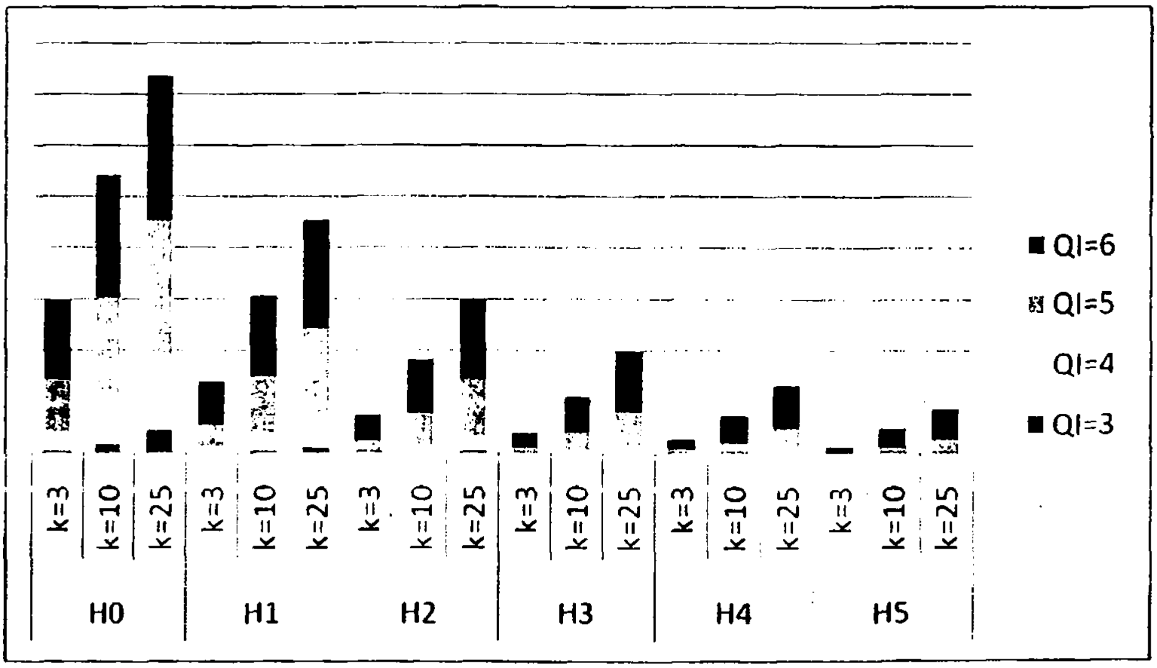


Figure 3.22 Relative volume of suppressed tuples for different combinations of generalization height, k and QI size (each sub-bar depicts the min number of suppressed tuples *fully* – i.e., not as a differential over the previous sub-bar; thus, it is meaningless to add the different values of sub-bars within a bar). Each vertical interval between horizontal lines corresponds to 10,000 tuples.

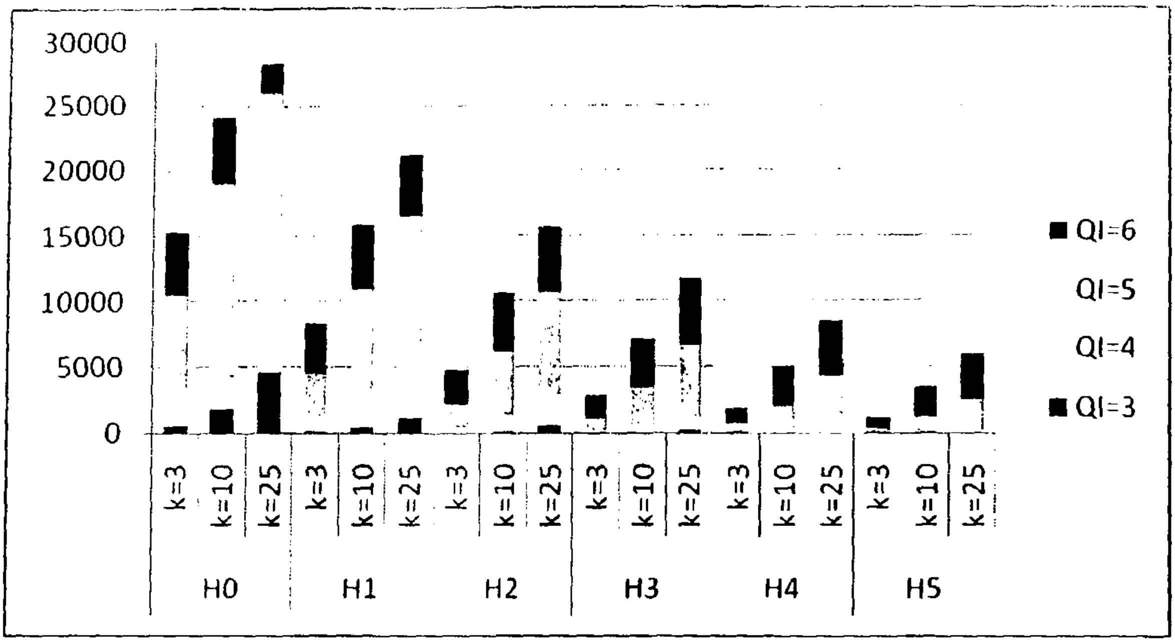


Figure 3.23 Relative volume of suppressed tuples for different combinations of generalization height, k and QI size (each sub-bar depicts the min number of suppressed tuples *incrementally* – i.e., with each bar as a differential over the previous sub-bar; thus, it makes sense to add the different values of sub-bars within a bar).

Our observations can be summarized as follows:

- A first observation (see for example Figure 3.23) is that all QI sizes have the same behavior: for low generalization levels they produce high numbers of suppressions and as the generalization level rises, the number of suppressed tuples drops gracefully. This is clearly depicted in Figures 3.20 for the average values and Figures 3.22 for the minimum values – in both these charts the absolute values of each QI size are depicted.
- The larger the QI size, the slower this drop is. This is evident as (a) in small heights, one can see QI=3 which quickly disappears then; (b) the increase to the suppressed tuples due to QI=6 is quite small compared, e.g., to QI=5 at lower levels; at higher levels however, the contribution of QI=5 drops whereas QI=6 that drops slower practically retains its contribution to suppression. See also Fig. 3.9 which clearly depicts the phenomenon.
- The increase of suppression due to the increase of k increases slowly with the height for as long as this has a meaning (see Table 3.5: within each column, as the height increases, the ratio of best solution for adjacent k's increases too). Interestingly, all k's fall with similar, but not identical speed as the height increases; see also Fig. 3.15 which clearly depicts the phenomenon.

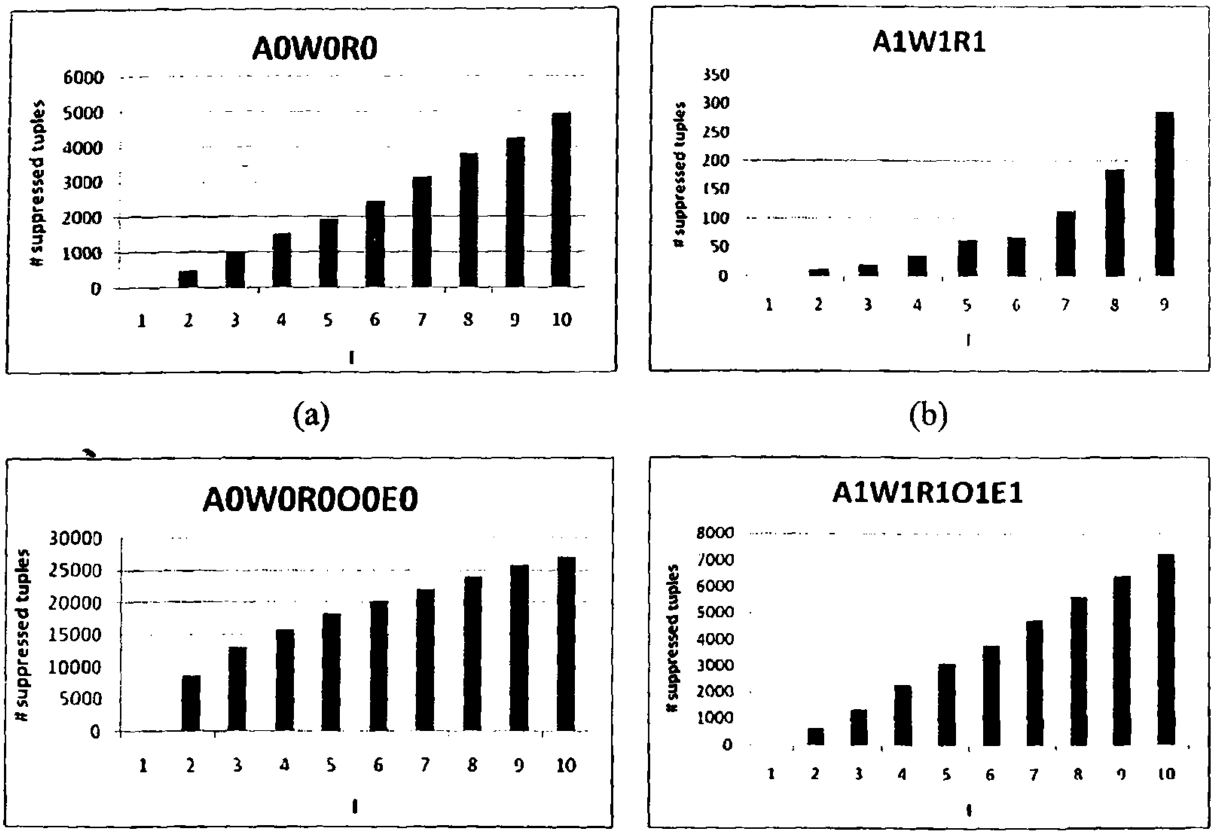
- When comparing average to minimum values it appears that the average values on suppression are practically 2,5 times as high as the minimum ones (observe Table 3.14). Again, this demonstrates clearly that it is worth paying the price to explore thoroughly the search space of possible solutions for a user's request.

As already mentioned, another result not clearly depicted in Figures 3.20 – 3.23 has to do with the particularities of each of the quasi-identifier attributes. Different attributes have different impacts to suppression; this will be detailed further in Section 5.

Table 3.14 Ratio of average number of suppressed tuples over minimum number of suppressed tuples for different QI sizes, values of k (in k-anonymity) and height.

	|QI|=3			|QI|=4			|QI|=5			|QI|=6			av
vg/min	k=3	K=10	k=25	k=3	k=10	k=25	k=3	k=10	k=25	k=3	k=10	k=25	
H0	1	1	1	1	1	1	1	1	1	1	1	1	1.0
H1	1.7	2.0	2.2	1.8	1.8	1.7	1.7	1.5	1.4	1.5	1.4	1.2	1.6
H2	2.0	2.1	1.9	2.7	2.6	2.5	2.6	2.1	1.8	2.1	1.8	1.5	2.1
H3	2.0	2.9	2.1	3.2	3.1	3.5	3.3	2.8	2.3	2.8	2.3	1.9	2.6
H4	2.1	1.6	2.3	4.3	3.8	3.1	3.2	3.3	2.8	3.1	2.6	2.2	2.8
H5	4.0	9.5	3.5	3.6	4.5	4.7	4.0	3.8	3.5	3.6	2.9	2.6	4.1
H6			1.5	3.8	2.6	8.5	6.0	4.4	4.4	4.5	4.2	3.5	4.3
avg	2.13	3.18	2.07	2.91	2.77	3.57	3.11	2.70	2.46	2.66	2.31	1.99	**2.6**

- 3.3. L-diversity for the Adult data set



(a) (b)

(c) (d)

Figure 3.24 Cumulative histogram για l-diversity

3.3.1. Comparison of different levels of generalization with fixed k and QI size

As one can clearly see in all the charts and tables of this section, as the height increases, the number of suppressed tuples drops with a high rate – after a certain level, it becomes meaningless to climb further up the lattice. The same phenomenon has been observed in the case of k-anonymity, too.

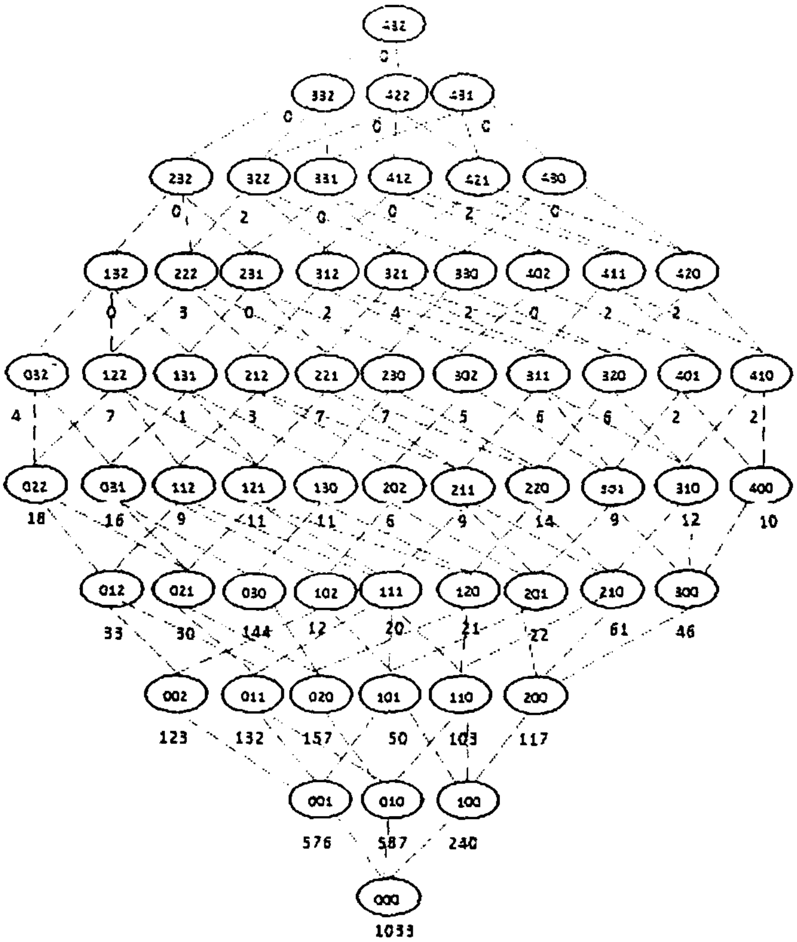


Figure 3.25 Full lattice with suppressed tuples for quasi identifier set of size 3. The QI is *Age*, *Work class*, *Race*.

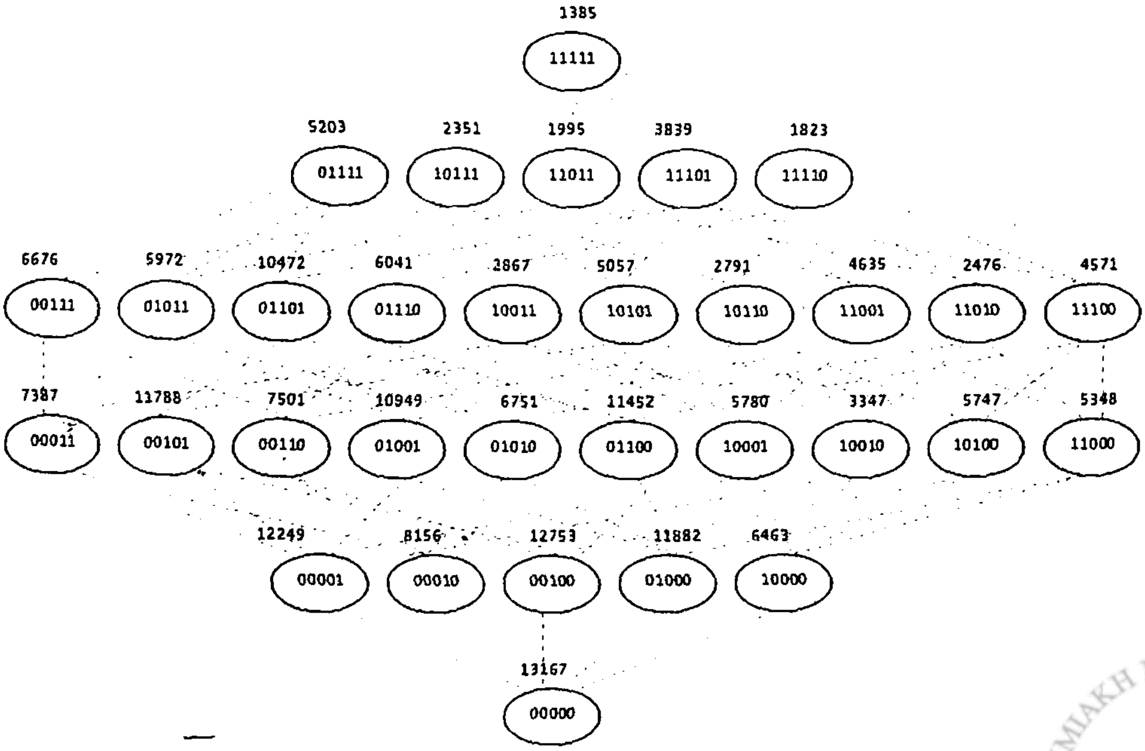


Figure 3.26 Full lattice with suppressed tuples for quasi identifier set of size 5. The QI is *Age*, *Work class*, *Race*, *Occupation*, *Education*

- *Not all attributes are born equal.* Again, as in the case of k-anonymity, it is clear that the careful choice of generalization scheme can significantly improve the number of suppressed tuples – especially at the lower parts of the lattices (that present the most important regions too).
    - Observe, for example, level H3 for QI=3. Node 102 has the smallest number of suppressed tuples (12). It is interesting to notice its parents at level H2: node 002 suppressed 123 tuples (much more than node 102), whereas node 101 suppressed 50. In other words, the best node is produced by (a) generalizing attribute *Age*, (b) not touching attribute *Work Class* and (c) slightly ascending over attribute *Race*. At the same time, the maximum number of suppressed tuples at level H3 is attained by node 030, which does the exact opposite of node 102: it only generalizes (a lot, at level 3) attribute *Work Class*.
    - At the same time, at level H3 for the partial lattice of QI=5 (*Age*, *Work class*, *Race*, *Occupation*, *Education*) we can observe that (again) the nodes with the largest and smallest number of suppressed tuples are practically complementary: node 01101 yields a suppression of 10472 tuples and node 11010 yields a suppression of 2476 tuples. The latter is produced by the node 10010 at H2 which is also the one with the smallest amount of suppressed tuples at its level. Clearly, the combination of the generalization of *Age* and *Occupation* minimize the suppression (see also the rest of the nodes of H3 that are produced by 10010: they have similar amounts of suppressed tuples and they are significantly lower than the other nodes of the level). Interestingly, the best generalization scheme at level H3, is not depicted in the partial lattice and it is 10020 (which practically says that occupation is fully generalized at its topmost level).

3.3.2. Comparison of different values of l and height with a fixed QI size

In this subsection, we report on our findings when comparing different values of *l* for the privacy criterion of l-diversity for their effect on the number of suppressed tuples. For each layer of nodes we list the minimum and average numbers of suppressed tuples, for QI sizes of 3 (Table 3.16) and 5. For the latter we explore the case of full lattice (Table 3.17). The results are also graphically depicted in Figure 3.27 and Figure 3.28 for QI=3 and Figures 3.29 and Figure 3.30 for QI=5.

Our observations follow closely the respective observations for k-anonymity and can be summarized as follows:

- As *l* increases, so does the amount of suppressed values (for the same height and QI size). The amount of suppression is not directly analogous to the value of *l*, however the scaling of the suppression is quite close to the scaling of the value of *l*.
- All the lines in all the charts of this subsection expose the same trend: as the height increases, the number of suppressed tuples drops quite quickly
- As in the case of k-anonymity, the ratio of minimum to average value is higher than 50% (Table 3.15) (in fact it rises to quite large values at big heights; if one removes the outliers the average ratio of average to minimum value is around 3).

Table 3.15 Ratio of average number of suppressed tuples over minimum number of suppressed tuples for different QI sizes, height and l-3

avg/min (l-3)				
	|QI|=3	|QI|=4	|QI|=5	|QI|=6
H0	1	1	1	1
H1	1,94	1,58	1,59	1,44
H2	2,27	2,58	2,29	1,92
H3	3,6	3,12	3,07	2,49
H4	1,9	5,29	3,17	2,83
H5	4,5	7,41	3,73	3,28
H6		6,17	4,78	4,03
H7		8,3	10,9	4,40
H8		2,1	10,69	5,61
H9		0,4	37,35	11,38

Table 3.16 Average and minimum number of suppressed tuples pre level for QI=3 over the full lattice for different values of l

|QI|=3 (full lattice)						
	l-3		l-6		l-9	
	min	avg	min	avg	min	avg
H0	1033	1033	2476	2476	4251	4251
H1	240	468	788	1430	1258	2356
H2	50	114	357	535	680	1160
H3	12	43	54	182	104	377
H4	6	11	22	50	29	99
H5	1	5	2	19	2	28
H6	0	2	0	10	0	11
H7	0	1	0	2	0	3
H8	0	0	0	0	0	0
H9	0	0	0	0	0	0

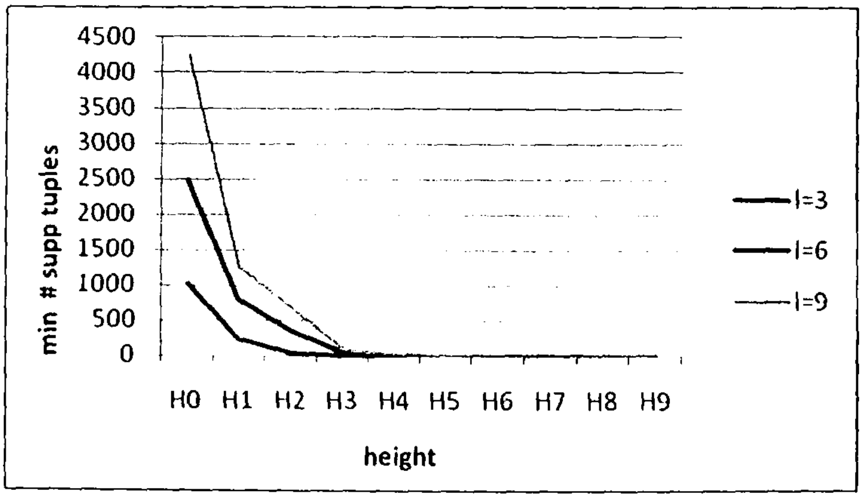


Figure 3.27 Minimum number of suppressed tuples pre level for QI=3 over the full lattice for different values of l

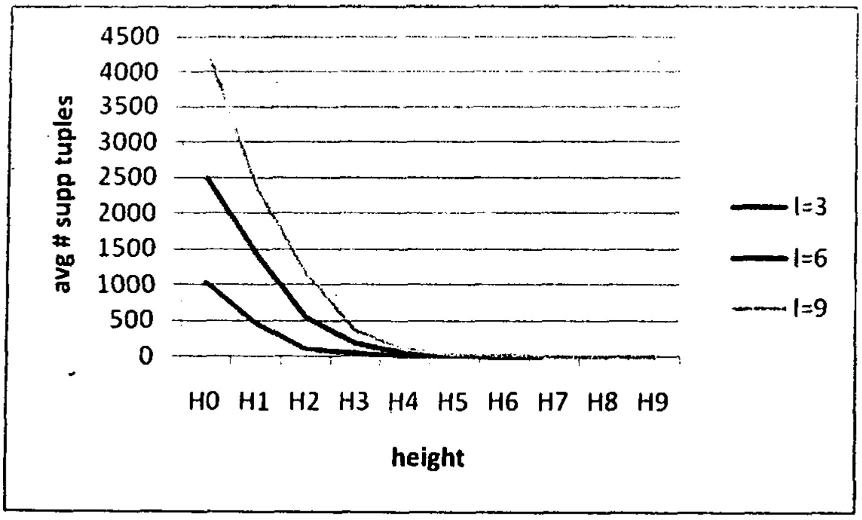


Figure 3.28 Minimum number of suppressed tuples pre level for QI=3 over the full lattice for different values of *l*

Table 3.17 Average and minimum number of suppressed tuples pre level for QI=5 over the full lattice for different values of l

|QI|=5 (full lattice)						
	l-3		l-6		l-9	
	min	**avg**	**min**	**avg**	**min**	**avg**
H0	13167	13167	20261	20261	25901	25901
H1	6463	10301	11971	17150	15624	22002
H2	3347	7694	6923	13890	10027	18551
H3	1774	5458	4043	10705	6392	15042
H4	1132	3594	2668	7738	4173	11506
H5	581	2172	1463	5180	2573	8181
H6	244	1168	689	3130	1241	5259
H7	50	545	302	1670	675	3015
H8	20	214	63	741	250	1466
H9	2	75	7	272	63	587
H10	0	23	0	85	14	196
H11	0	7	0	26	0	57
H12	0	2	0	9	0	15
H13	0	0	0	2	0	4
H14	0	0	0	0	0	0
H15	0	0	0	0	0	0

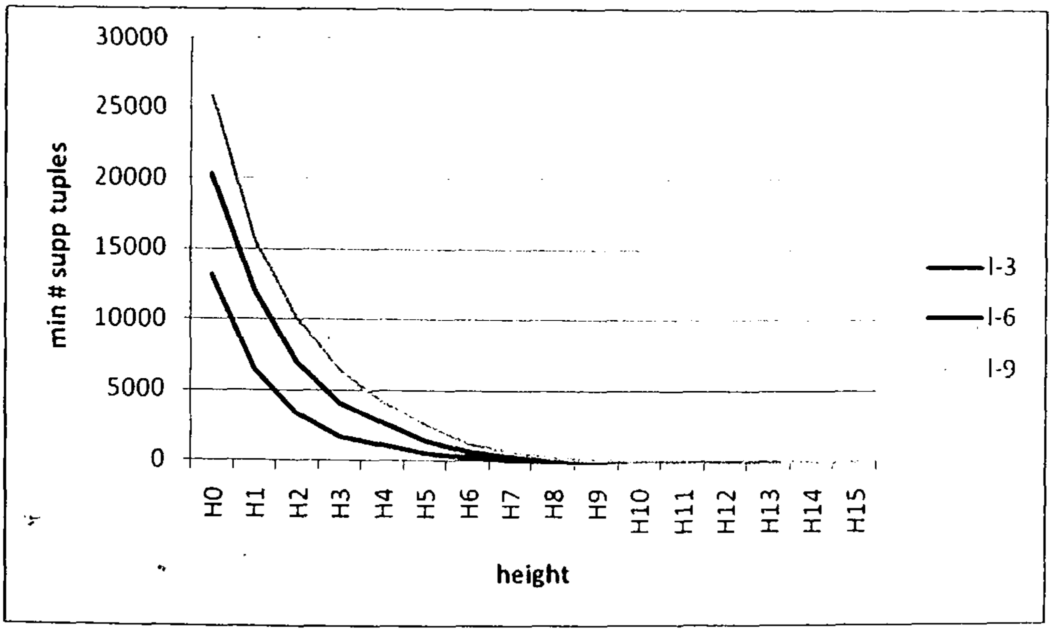


Figure 3.29 Minimum number of suppressed tuples pre level for QI=5 over the full lattice for different values of l

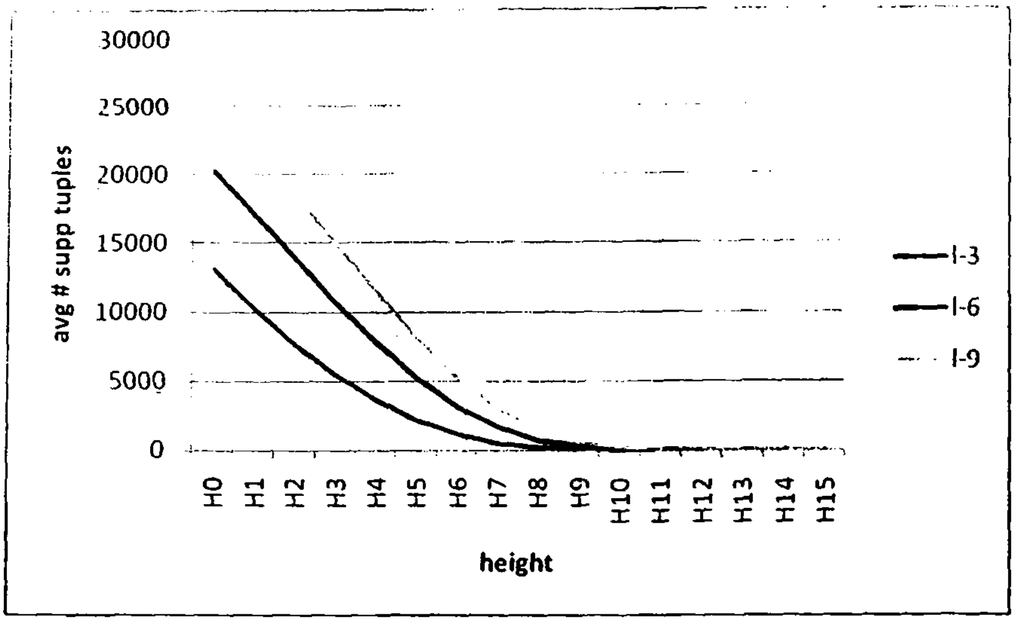


Figure 3.30 Average number of suppressed tuples pre level for QI=5 over the full lattice for different values of l

3.3.3. Comparison of different QI sizes (over different levels) with a fixed value of l

In this subsection, we focus our observations in the effect of increasing the QI size over the amount of suppressed tuples. We fix the level of *l*-diversity to *l* = 3 and present our results per different levels of generalization and QI size.

Our observations can be summarized as follows:

- Clearly, different QI sizes at the same level have varying levels of increase to the minimum number of suppressed tuples. This scale up can range from 5 to 2 and systematically decreases as QI increases. However, as in the case of k-anonymity, it is clear again that the size of QI is the determining factor for the amount of suppression that can take place.
- Moreover, it is clear that statistically tolerable amounts of suppressed tuples are attained slower as the size of QI grows. For example, the suppression percentage falls under 1% of the total volume of data at height H1 for QI = 3, H3 for QI = 4, H6 for QI = 5 and after H8 for QI = 6 (all at the same level with k-anonymity).

As in the case of k-anonymity, we also observe that a QI of size n drops to the levels of suppression of the QI of size n-1 around 2-3 levels of generalization later

Table 3.18 Average and minimum number of suppressed tuples per level for all QI sizes and l=3 over the full lattice

	|QI|=3		|QI|=4		|QI|=5		|QI|=6	
	avg	**min**	**avg**	**min**	**avg**	**min**	**avg**	**min**
H0	1033,0	1033	5116,0	5116	13167,0	13167	17871,0	17871
H1	467,7	240	3118,0	1972	10300,6	6463	15405,8	10671
H2	113,7	50	1644,4	637	7694,4	3347	12928,4	6719
H3	43,2	12	750,1	240	5457,6	1774	10518,6	4210
H4	11,4	6	264,8	50	3593,5	1132	8200,0	2894
H5	4,5	1	89,0	12	2171,9	581	6079,5	1848
H6	1,7	0	24,7	4	1168,2	244	4236,2	1049
H7	0,7	0	8,3	0	545,0	50	2740,5	622
H8	0,0	0	2,1	0	213,9	20	1617,6	288
H9	0,0	0	0,4	0	74,7	2	854,0	75

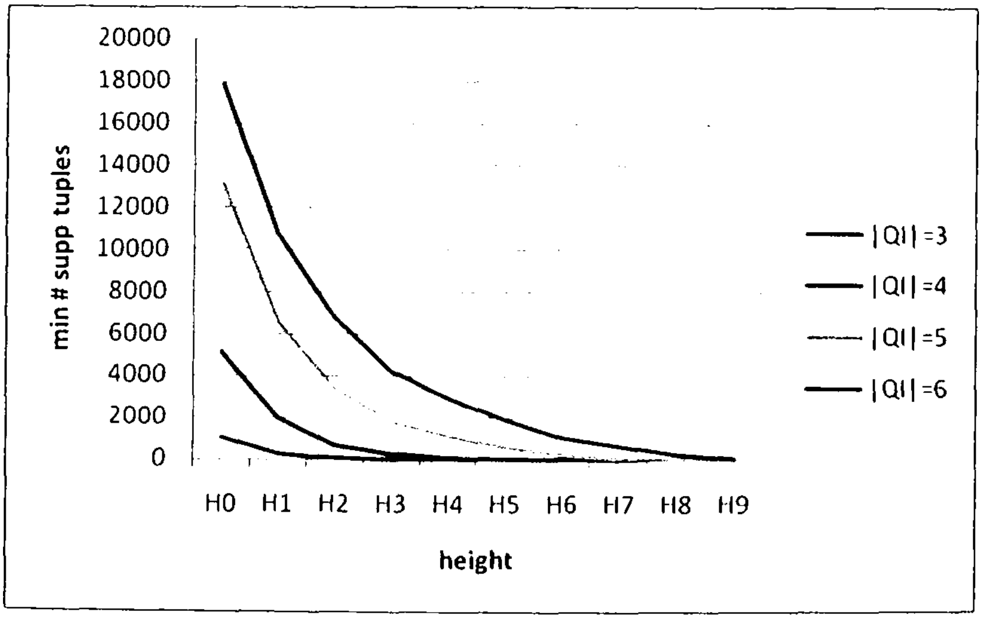
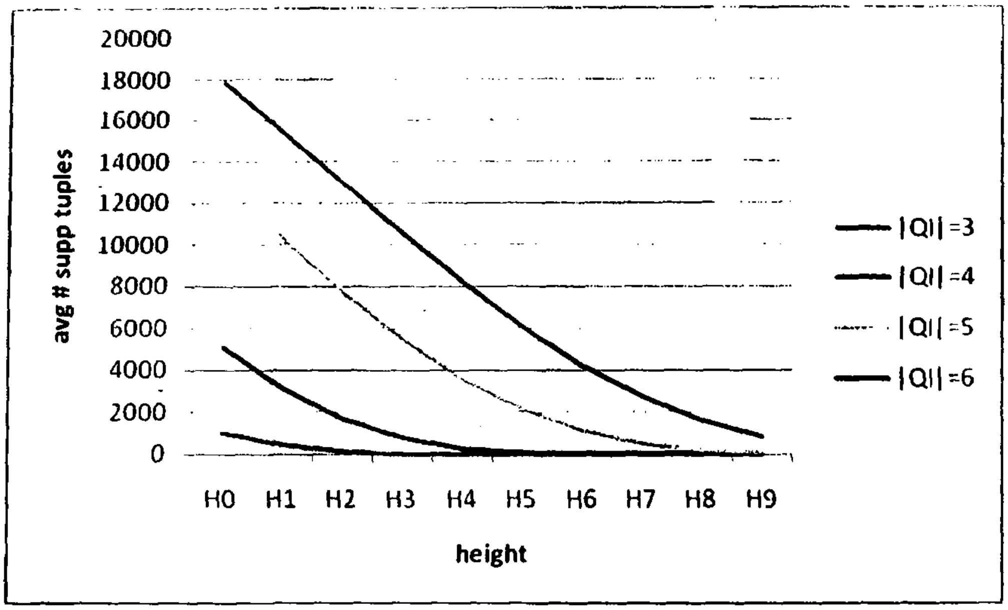


Figure 3.31 Minimum number of suppressed tuples per level for all QI sizes and l=3 over the full lattice



Figuŗe 3.32 Average number of suppressed tuples per level for all QI sizes and l=3 over the full lattice

3.4. K-anonymity and L-diversity for the PUMS data set

In this subsection, we report on our findings with the PUMS data set [IPUMS]. The PUMS data set comprises 600000 records of the USA census. The attributes of the PUMS data set are *age, birthplace, education, gender*. The hierarchies of these attributes are the same for the same with Adult for the attributes *age, education and gender* but different for the *birthplace* (we can see birthplace hierarchy in Figure 3.33).We have used *occupation* as the sensitive attribute for this data set.

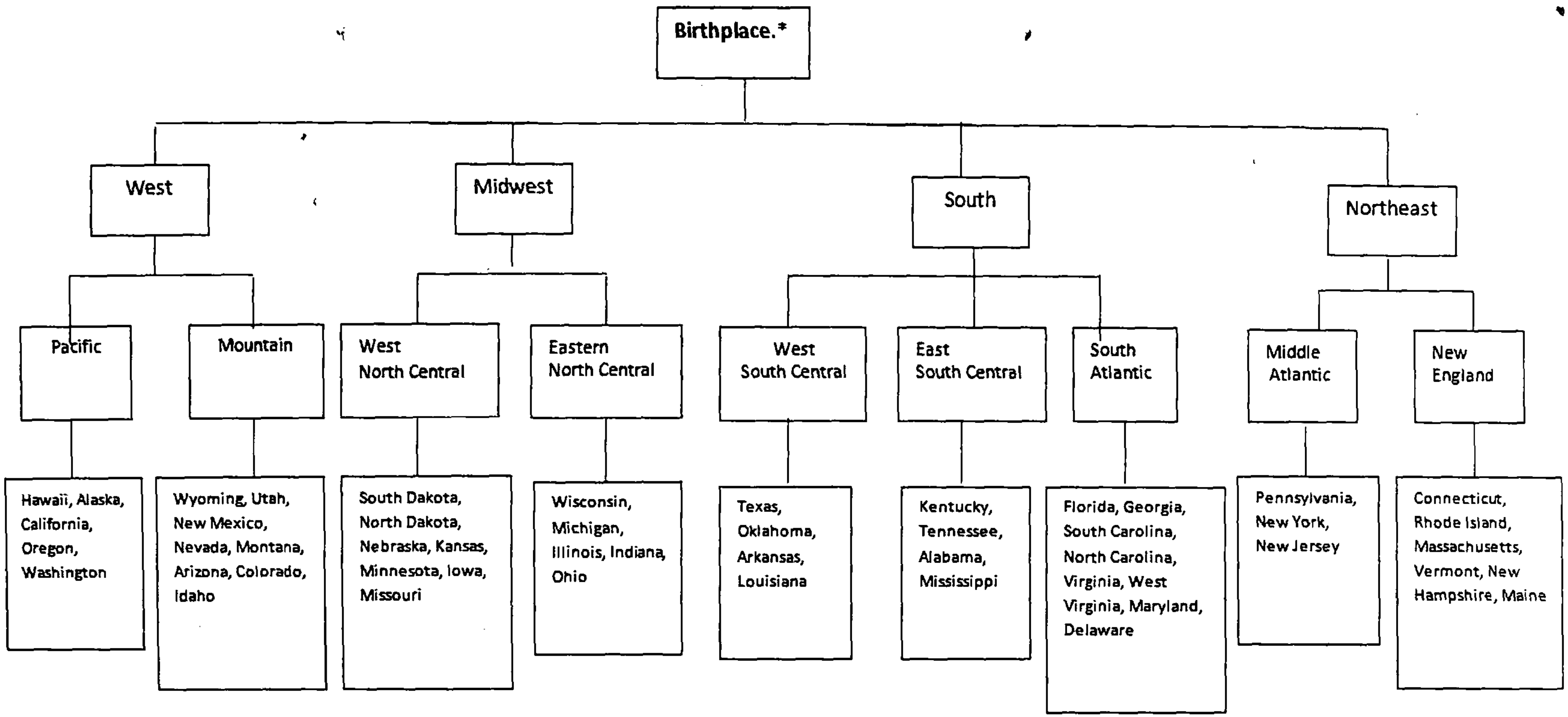

Birthplace.*
West
Midwest
South
Northeast
Pacific
Mountain
West North Central
Eastern North Central
West South Central
East South Central
South Atlantic
Middle Atlantic
New England
Hawaii, Alaska, California, Oregon, Washington
Wyoming, Utah, New Mexico, Nevada, Montana, Arizona, Colorado, Idaho
South Dakota, North Dakota, Nebraska, Kansas, Minnesota, Iowa, Missouri
Wisconsin, Michigan, Illinois, Indiana, Ohio
Texas, Oklahoma, Arkansas, Louisiana
Kentucky, Tennessee, Alabama, Mississippi
Florida, Georgia, South Carolina, North Carolina, Virginia, West Virginia, Maryland, Delaware
Pennsylvania, New York, New Jersey
Connecticut, Rhode Island, Massachusetts, Vermont, New Hampshire, Maine

Figure 3.33 The hierarchy for the *Birthplace* dimension

K-anonymity. Although the data set size is significantly larger than the one of Adult, the quasi-identifier size of the PUMS data set is small, as it comprises only 4 attributes. Therefore we have not experimented with the quasi-identifier size, but rather we have explored the interrelationship of k with the different heights of the lattice. In Table 3.19 and Figure 3.34 we depict the average number of suppressed tuples per height and in Figure 3.35 and Table 3.20 we depict the minimum number of suppressed tuples per level.

Our observations can be summarized as follows:

- The general trend of suppression as the height increases is quite similar with the one discovered at the Adult data set: the suppression levels are high for small heights and quickly drop to small amounts of suppressed tuples. This holds for all the values of k that we have tested (i.e., k=3, 10, 50, 100, 150). Interestingly, all the values of k demonstrate this cut-off behavior within the range of two heights (H3 and H4) when the minimum number of suppressed tuples is concerned (see Figure 3.4.2).

Table 3.19 Average number of suppressed tuples for different values of k for the PUMS data set

	Average # of suppressed tuples				
	k=3	**k=10**	**k=50**	**k=100**	**k=150**
H0	31933.0	128493.0	369177.0	490040.0	539066.0
H1	12020.3	57561.5	220812.5	330966.5	396465.0
H2	5071.2	25413.6	124158.2	204604.8	260322.4
H3	1790.9	9873.9	59787.8	115041.2	158743.6
H4	530.4	3197.8	23421.8	50488.1	77780.2
H5	149.4	1028.2	7789.6	17577.8	28101.9
H6	31.8	290.3	2307.0	5288.1	8920.8
H7	4.5	54.9	679.8	1534.8	2667.9
H8	0.6	6.3	147.9	402.4	663.5
H9	0.0	0.0	9.4	79.9	156.5
H10	0.0	0.0	0.0	0.0	15.4
H11	0.0	0.0	0.0	0.0	0.0
H12	0.0	0.0	0.0	0.0	0.0

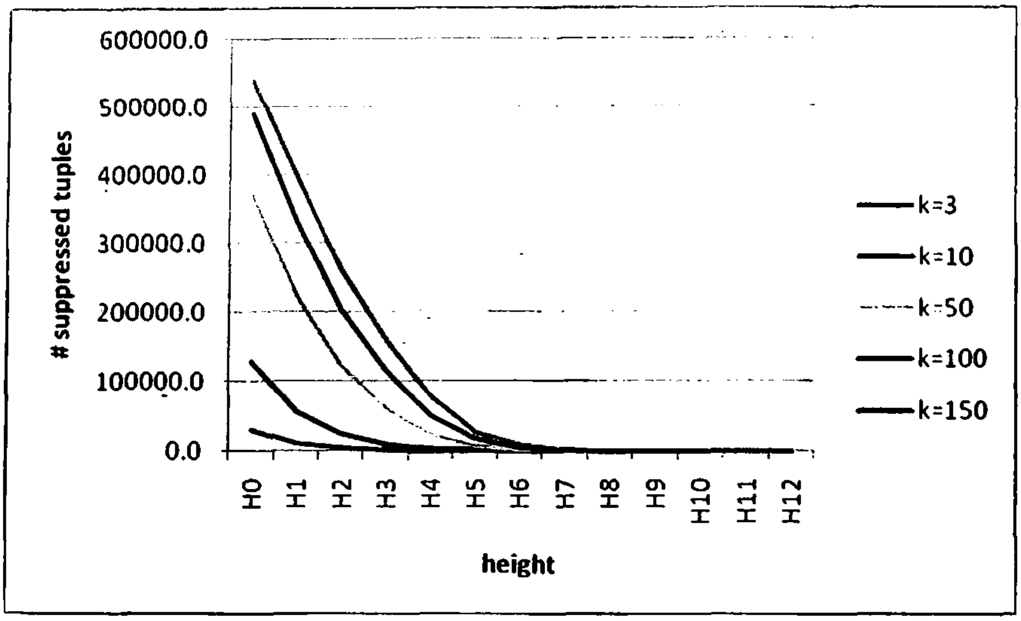


Figure 3.34 Average number of suppressed tuples for different values of k for the PUMS data set

- The relationship between minimum and average suppressed tuples is quite different than the case of the Adult data set. The minimum values drop very quickly with the increase of the height, whereas this fall is much slower than the case of the average suppression: in other words, the choice of a good anonymization scheme is much more important in the case of the PUMS data set.

Table 3.20 Minimum number of suppressed tuples for different values of k for the PUMS data set

	min # of suppressed tuples				
	k=3	**k=10**	**k=50**	**k=100**	**k=150**
H0	31933	128493	369177	490040	539066
H1	4462	27036	140719	226622	288288
H2	570	3778	30578	64388	95620
H3	169	1370	11141	27287	42793
H4	3	197	2037	4775	7705
H5	0	18	527	1193	2482
H6	0	0	95	391	779
H7	0	0	0	0	0
H8	0	0	0	0	0
H9	0	0	0	0	0
H10	0	0	0	0	0
H11	0	0	0	0	0
H12	0	0	0	0	0

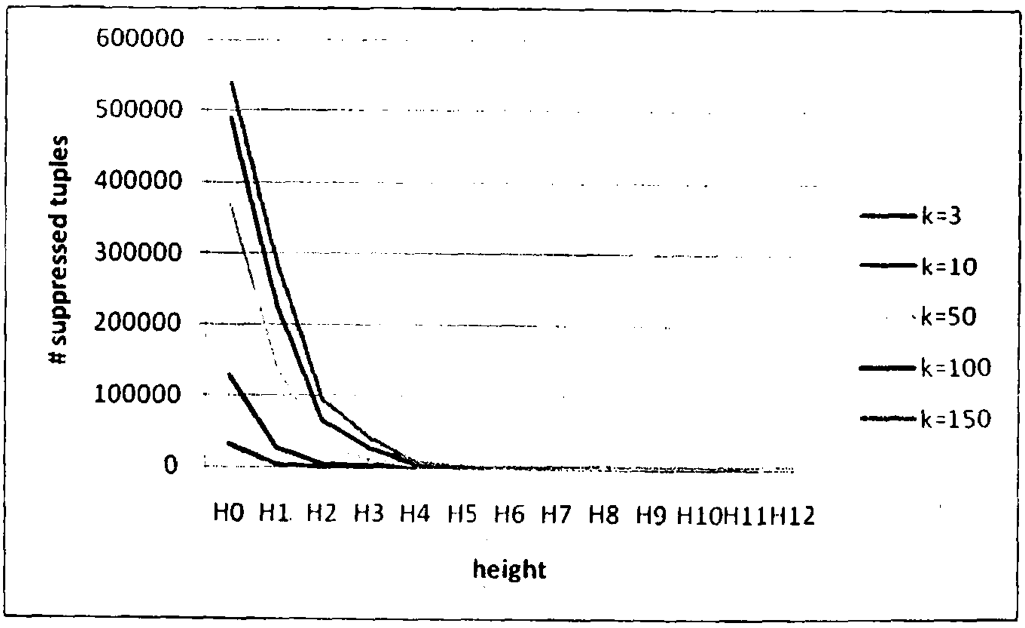


Figure 3.35 Minimum number of suppressed tuples for different values of k for the PUMS data set

L-diversity. We have tested the PUMS data set for its behaviour concerning the suppression of tuples in the case of *l*-diversity for different values of *l* (specifically, 3, 6, 9). The choice of values for *l* was such that the data set was not massively suppressed for reasonable heights (observe Table 3.22, where H2 still holds around 1% of suppression for the best possible solution). We depict our findings in Table

3.21 and Figure 3.36 for the average number of suppressed tuples per height and value of *l* as well as in Table 3.22 and Figure 3.37 for the minimum number of suppressed tuples – i.e., the best possible solution—per height and value of *l*.

Table 3.21 Average number of suppressed tuples for different values of l for the PUMS data set

	avg # of suppressed tuples		
	l=3	l=6	l=9
H0	37187.0	96136.0	147964.0
H1	14255.8	41891.5	71999.5
H2	6136.4	18788.9	33843.0
H3	2177.5	7380.6	14116.6
H4	630.1	2329.0	4849.2
H5	178.1	721.6	1524.5
H6	38.1	183.8	438.3
H7	5.0	28.7	84.0
H8	0.6	2.8	9.4
H9	0.0	0.0	0.9
H10	0.0	0.0	0.0
H11	0.0	0.0	0.0
H12	0.0	0.0	0.0

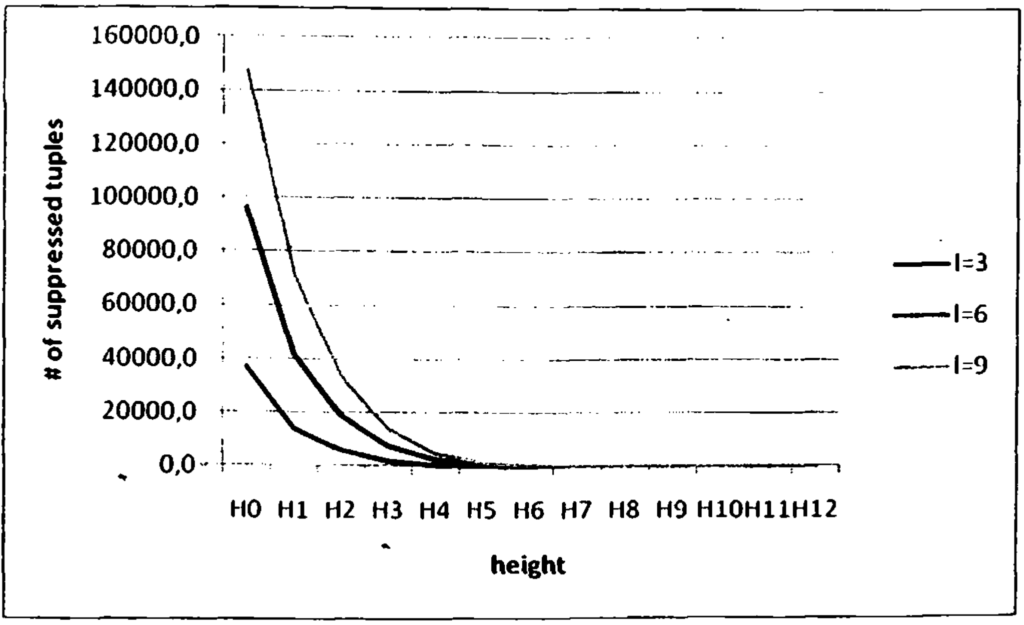


Figure 3.36 Average number of suppressed tuples for different values of l for the PUMS data set

Table 3.22 Minimum number of suppressed tuples for different values of l for the PUMS data set

	min # of suppressed tuples		
	l=3	l=6	l=9
H0	37187	96136	147964
H1	5333	20837	40589
H2	701	2779	6129
H3	204	927	2026
H4	3	89	332
H5	0	7	41
H6	0	0	0
H7	0	0	0
H8	0	0	0
H9	0	0	0
H10	0	0	0
H11	0	0	0
H12	0	0	0

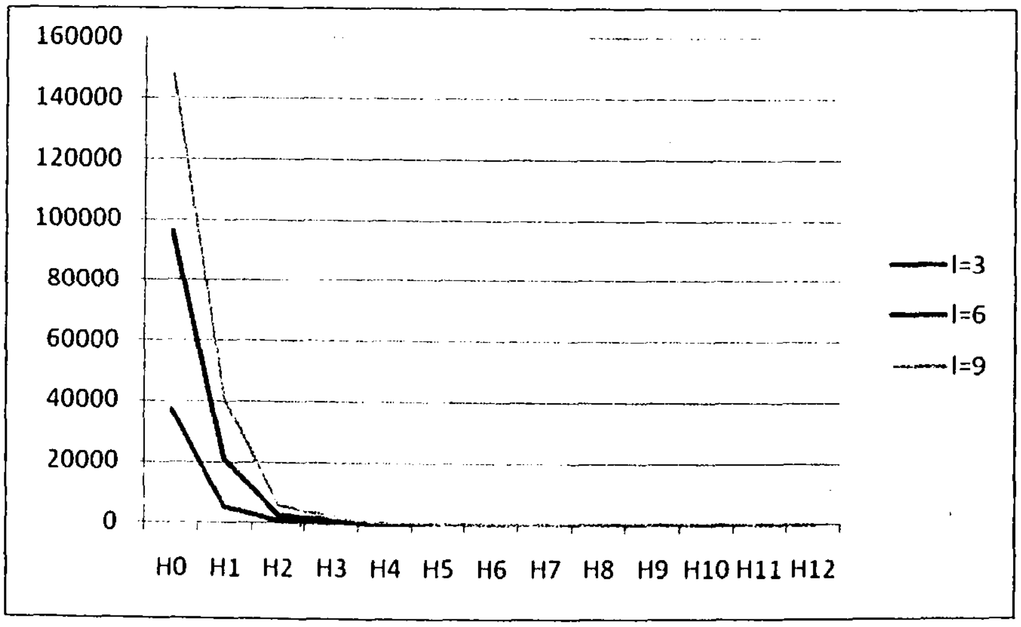


Figure 3.37 Minimum number of suppressed tuples for different values of l for the PUMS data set

Our findings can be summarized as follows:

- The general tendency for the drop of suppressed tuples as the height increases is verified once again: there is an exponential drop of the suppression as the height increases, for all values of *l*.
- The cut-off point, whereas suppression becomes acceptable low for the best possible anonymization scheme (Figure 3.37) is quite low (around H1 and H2 for all values of l), whereas this picture is quite different for the average case (Figure 3.36) where it is found approximately two levels higher.
- The comparison of k-anonymity and *l*-diversity shows a remarkable resemblance for the general trend and the behavior of the amount of suppressed tuples as the height or the privacy criterion increase their value. Again, we can think of k-anonymity as a good estimator of *l*-diversity.

3.5. The price of histograms

The lattice of generalization schemes and most importantly, the histograms with which the lattice is annotated come with a price, both in terms of space and in terms of construction time. In this section, we discuss these preprocessing and storage prices.

K-anonymity. In Figure 3.38 we depict the time needed to construct the full lattice and to annotate it with the necessary histograms for the case of k-anonymity. Naturally, the latter task takes up practically all the necessary time. As the QI size increases the time also increases exponentially. However, for all the QI sizes that we have considered, the time ranges from few seconds to less than 20 minutes.

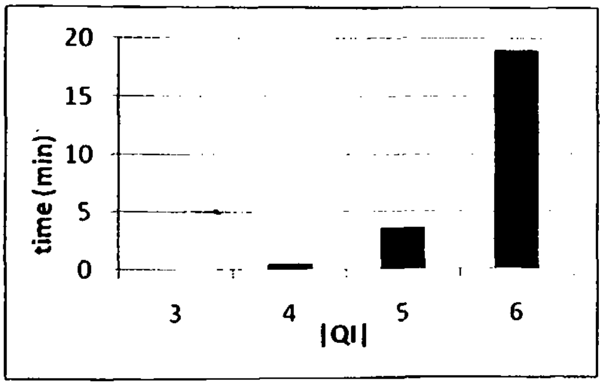


Figure 3.38 Construction time for the full lattice and its k-anonymity histograms for the Adult data set.

Clearly, the size of the data set influences the time needed to construct the full lattice's histograms. Remember that the histogram for each node in the lattice practically requires two aggregate queries over all the data set (one that constructs the groups and another that counts the group sizes frequencies). This does not explain, however, the exponential delay with the increase of QI size; the reason for this phenomenon is depicted in Figure 3.39, where we present the lattice size in terms of nodes and edges.

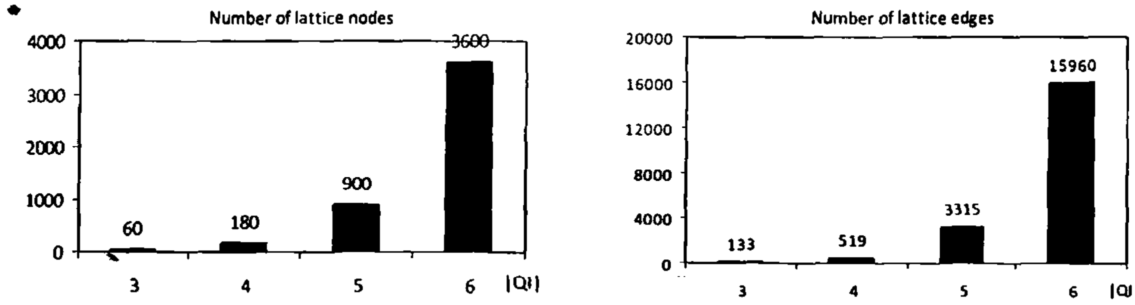


Figure 3.39 Lattice size in terms of nodes and edges for the k-anonymity lattice of the Adult data set.

Again, we can observe the same exponential increase (esp., in the case of edges).

Although the time spent to construct the lattice is significant, the amount of memory that is needed to keep the histograms in main memory is quite small. Observe Figure 3.40, where we depict the amount of main memory spent to retain the histograms for all the nodes of the lattice in the case of k-anonymity. Remember that the lattice size is dependent only upon the number of dimensions and the number of levels of each dimension and not upon the size of the data set (in fact, the data set influences the size of the histogram only with respect to the number of groups produced – however for each group we only need two integers, so this cost is not so important after all). Again, the increase is exponential in terms of the QI (which is clearly due to the exponential increase in the number of lattice nodes).

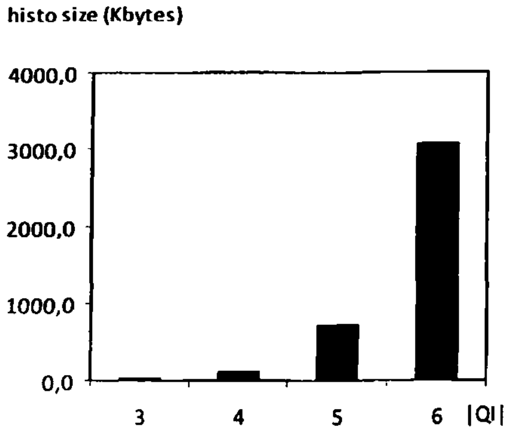


Figure 3.40 Main memory spent to retain the k-anonymity histograms for the Adult data set (KB).

L-diversity. In our experimentation with the Adult data set, we have also explored the case of L-diversity. Specifically, in Figure 3.41 we depict the construction time for the *l*-diversity histograms, and in Figure 3.42 we depict the memory cost for retaining these histograms. The observed phenomena are practically the same as with k-anonymity; however observe that the number of distinct values that *l* can take (the x-axis of the histogram, in other words) is much less than the respective values for the case of k-anonymity; therefore, the size needed is lower for *l*-diversity than k-anonymity.

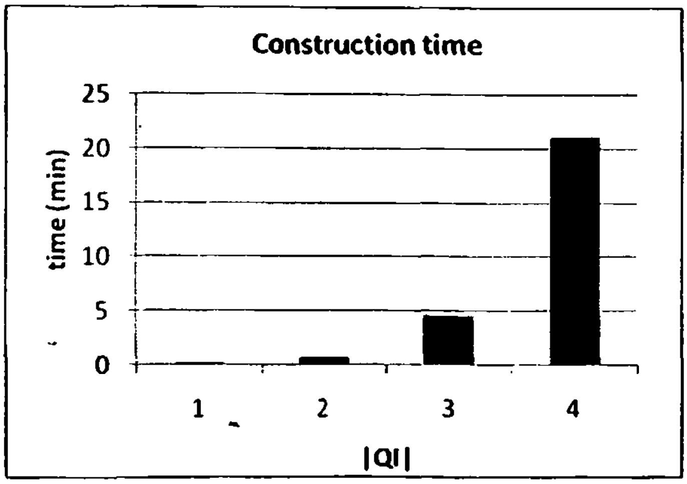


Figure 3.41 Construction time for the full lattice and its l-diversity histograms for the Adult data set.

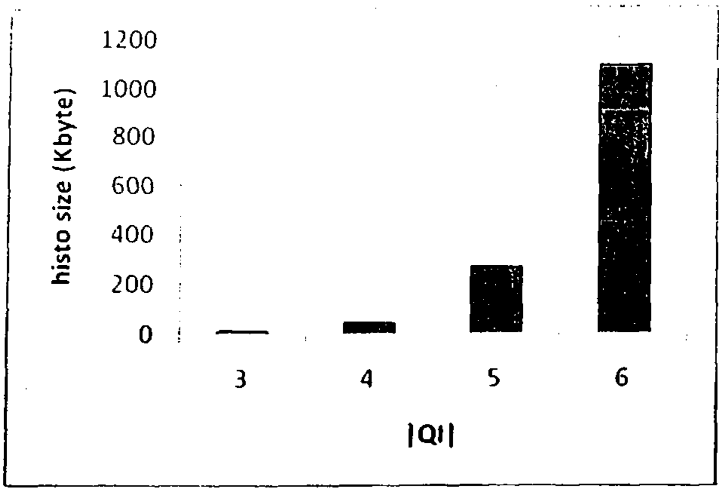


Figure 3.42 Main memory spent to retain the l-diversity histograms for the Adult data set (KB).

IPUMS. Apart from the Adult data set, the observed values are also consistent with the case of the IPUMS data set (see Figure 3.43). As we have already mentioned, the lattice size is practically independent from the data size and dependent mainly upon the lattice's hierarchies; therefore the histogram sizes are comparable for Adult with QI=5 and IPUMS (the hierarchies are slightly different). The construction time, however is quite different and this is clearly due to the fact that the size of IPUMS is 50 times the size of the Adult data set. This explains the difference in time costs.

	k-anonymity	**l-diversity**
Average time (minutes)	10,5	38,087
Histo size (Kbytes)	530,688	58,424

Figure 3.43 Construction time (min) and main memory spent (KB) for the IPUMS data set.

3.6. Summary of findings

The goal of this chapter has been to study the relationship of suppression, generalization height and privacy criterion and via this study, to characterize the importance of the problem. *Overall, we can safely claim that the problem is valid and important. Low generalization heights (that are of more interest to us due to their information utility), or large values for the privacy criterion (which is of more interest to us due to the increased privacy it offers to individuals), or erroneous choice of generalization scheme can result in large amounts of suppressed data, quite possibly* *much higher as compared to more careful choices concerning the generalization scheme.*

Our detailed findings concerning the relationship of the involved parameters can be summarized as follows:

- As the generalization height increases, the suppression drops quickly at small heights; the drop in suppression is less important in higher heights, where the number of suppressed tuples becomes statistically small and drops slowly. Interestingly, the overall trend for the decrease of suppression is practically the same for different values of *k* or *l* – of course, with different amounts of suppressed tuples.
- As the value for the privacy criterion (e.g., *k* in k-anonymity) increases, the suppression increases too. This is especially important in lower heights of generalization that are both important due to their information utility and demonstrate high volumes of suppression.
- As the size of the quasi identifier set increases, the effect to suppression is significant, as suppression increases too – sometimes drastically. Some quantitative evaluations around this theme suggest that (a) given a specific height and k an increase in QI size by one increases the suppression by a factor of 2 – 3; (b) to attain the same suppression threshold an increase in QI size by one, requires ascending 1-2 levels for k-anonymity and 2-3 levels for *l*-diversity.
- Not all attributes, generalization levels and, consequently, generalization schemes have the same effect to suppression. It is noteworthy that within the same height, the minimum possible suppression is approximately 2.5 times lower than the average for k-anonymity and 3 times lowers for l-diversity. This is especially evident in cases where the suppression has high values or values that cannot really be tolerated; on the other hand, for too large values of suppression (e.g., too large QIs or *k*) the relationship between average and minimum value does not follow this rule.
- Based on the above, it is important that for case that do matter, and where we can really attain good amounts for tuple suppression, it is really important to carefully pick the generalization scheme that will minimize this suppression.

- The faster we identify these generalization schemes the faster the process completes.

We should also note that the above findings seem consistent with both k-anonymity and l-diversity over two data sets – with slight variations of course. Also, we should mention here that the effect of QI size to lattice is really important (of exponential nature) and this mainly affects the construction time of the lattice's histograms (which is also affected by the database size, of course, however with lesser degree of importance)

CHAPTER 4. ONLINE NEGOTIATION ALGORITHMS FOR PUBLISHING PRIVATE DATA

In this section, we explore a reference algorithm for the on-line negotiation over antagonistic privacy criteria. The general idea of the algorithm is based on two steps: (a) an off-line, preprocessing step, where the histogram lattice is built and (b) an online step, where the users pose requests for anonymizations over different combinations of criteria and the algorithm returns either exact results or suggestions.

Preprocessing step

Input:

- a data set *T*, comprising an identifier attribute *ID*,
- a set of quasi-identifier attributes **QI** = $\{A_1, \ldots, A_n\}$,
- a sensitive attribute *S*,
- a set of generalization hierarchies **H** = $\{\mathbf{h}_1, \ldots, \mathbf{h}_n\}$, one for each quasi-identifier attribute
- a privacy constraint (e.g., *k*-anonymity, *l*-diversity, m-uniqueness, …),

Output:

- a histogram lattice L(V,E) such that: (a) a node *v*, labelled $[l_1, \ldots, l_n]$ exists for every combination of hierarchy levels l_i, over all quasi identifier hierarchies, (b) a set of edges stemming from every node $v = [l_1, \ldots, l_k, \ldots, l_n]$ to nodes u, with u being nodes of the form $[l_1, \ldots, l_{k+1}, \ldots, l_n]$, for all k = 1, …, n, (c) a histogram **C** with pairs of the form [*statProp*, *counter*] annotates every node v, with *statProp* being the statistical property of a group that determines the privacy level and *counter* being the number of groups with size *groupSize* in the result of this grouping query.

Figure 4.1 Off-line preprocessing step

The generic pre-processing step, where the lattice is built and each of its nodes is annotated with the appropriate histogram is depicted in Figure 4.1. The only unclear point to the above definition is the statistical property parameter, which we clarify right away. The problem is defined for privacy criteria that can be defined as properties of each group. Remember that given a node *v* $[l_1, \ldots, l_n]$, its groups are formed when we group *T* by the values of $[l_1, \ldots, l_n]$. Then, a statistical property is tested for every group, depending on the privacy criterion. For example, the privacy criterion of k-anonymity requires that each of these groups accounts for at least k tuples; the criterion of *l*-diversity requires at least *l* different sensitive values in the group, and so on. This statistical property is counted in each histogram. So, for example, a value of <45, 67> in *l*-diversity means that there are 67 groups with 45 different sensitive values for a certain generalization scheme. Other statistical properties of this nature include the entropy of a group (in entropy-based *l*-diversity) of the distance of the distribution of the sensitive values of the group to the distribution of the sensitive values of the data set (in t-closeness).

The "pluggable" parameters of the problem of on-line privacy negotiation can be summarized as follows:

Parameters

Table 4.1 Problem parameters and possible examples

Lattice	**Possible values**
Lattice extent	Full or Partial lattice
Lattice construction	Offline or on-line
Lattice contents	Depends on the privacy constraint(s) supported. Example: histogram's <X-value, Y-value> are <groupSize, counter> for k-anonymity
Algorithm	
Privacy constraint	k-anonymity, l-diversity, t-closeness, m-uniqueness, …
QoS()	A utility function that determines the best solution (including tie-breakers). Examples: Height of a solution, discernibility, ...

Once the offline, pre-processing step, is completed, then we are ready to exploit the lattice in order to devise anonymization schemes in ab on-line fashion. The problem specification as a set of input/output specification for the generic case is depicted in Figure 4.2.

On-line step

Input:

- a histogram lattice L(V,E) over a data set *T* and a set of hierarchies **H** as before
- a privacy constraint (e.g., *k*-anonymity, *l*-diversity, m-uniqueness, ...),
- fixed constraints for
  - (d1) the maximum height per attribute that the anonymization method can attain $\mathbf{h} = [h_1^c, \ldots, h_n^c]$,
  - (d2) the lowest value for the privacy constraint (e.g., *k* for *k*-anonymity) and
  - (d3) the maximum number of suppressed tuples that the user is willing to tolerate *MaxSupp,*
- a quality criterion function *QoS*() for the assessment of the best possible anonymization when more than one answers are available (e.g., the solution with the lowest height, and possibly the less suppressed tuples, or maximum discernibility, as another example).

Output:

- An anonymized data set *T** such that
  - *T** is a generalization of *T*, *T** fulfils the abovementioned privacy constraints (d1) – (d3), and, *T** minimizes the quality criterion function *QoS*(*T**), if such a *T** can be attained,

or,

  - A set of alternative generalizations that are also generalizations of *T* and each of them minimizes the deviation for one of the parameters of the problem, specifically, (a) the acceptable generalization heights, (b) the minimum acceptable value for the privacy constraint and (c) the number of suppressed tuples.

Figure 4.2 Problem specification for the generic case of on-line privacy negotiation

4.1. Simple negotiation for k-anonymity (as privacy criterion) and height (as the criterion for the quality of the solution)

In the sequel we present a simple algorithm to perform on-line negotiation over conflicting privacy requirements. The following table shows how the parameters of the generic problem are instantiated for the problem under consideration.

Table 4.2 Parameters of the Algorithm

Offline	Used value
Lattice	Full lattice construction
<X-value, Y-value>	groupSize for k-anonymity, counter
On-line	
Privacy constraint	k-anonymity
QoS()	Height of a solution

Algorithm *SimpleAnonymiyNegotiation* operates over a relation *R* with a hierarchy *H* that results in a lattice annotated with histograms *L*. In the rest of our deliberations we will focus on the case of k-anonymity, however the same algorithm applies to the case of l-diversity, with the histograms of the lattice *L* and the constraints checking for determining whether a candidate node of the lattice is actually a solution being the only differences among the two cases. The proposed algorithm takes as input a table to be generalized, a set of hierarchies for the quasi-identifier attributes, the histogram lattice for all possible combinations of the generalization levels, and the requirements for the maximum desirable generalization level per quasi-identifier, the maximum tolerable number of tuples to be suppressed and the least size of a group, *k*, as the privacy constraint. The output of the algorithm are either

(a) an node of the lattice (i.e., a generalization scheme) that provides the best possible *exact* solution to the user requirements (with best possible being interpreted as the one with the lowest height, and, if more than one candidate solutions have this lowest height, the one with the minimum suppression), or,

(b) three suggestions for approximate answers to the user request, the first relaxing the number of suppressed tuples, the second relaxes the constraints on the heights per dimension and the third relaxing the minimum acceptable privacy criterion (e.g., *k* in k-anonymity).

The algorithm proceeds as follows:

```
Algorithm SimpleAnonymityNegotiation(L,k,h,MaxSupp)
In: Lattice L with the histograms for R,H, constraints for k, h, MaxSupp
Out: an exact solution s[v,k,h,supp] or s1,s2,s3, si=[v_i,k_i,h_i,supp_i]
Var: a 2D vector of candidate solutions Candidates[hmax][]
Begin
Let v_max be the node that corresponds to the constraint h ;
if v_max is visited then exit;
mark v_max as visited;
if (checkExactSolution(v_max,L,k,h,MaxSupp) == true){
        Candidates[height(v_max)] = Candidates[height(v_max)] ∪ {v_max};
        for all v_c in lower(v_max)

        ExactSublatticeSearch(v_c,L,k,h,MaxSupp,Candidates);
//when the recursion is over, the Candidates has the full list of nodes
//that can serve as candidate solutions
        minHeight = minimum height having Candidates[minHeight] != {};
        v_win = v in Candidates[minHeight] with the lowest possible suppression for k;
        return(v_win,k,minHeight,suppressed(v_win,k));
}
else{
        approxSol_1 = ApproximateMaxSupp(L,v_max,k,h,MaxSupp);
        approxSol_2=ApproximateH(L,v_max,height(v_max),height(top),k,h,MaxSupp);
        approxSol_3 = ApproximateK(L,v_max,k,h,MaxSupp);

        return approxSol_1, approxSol_2, approxSol_3;
}
End.
```

Figure 4.3 Algorithm Simple Anonymity Negotiation

First the algorithm identifies a reference node in the lattice, to which we refer a v_{max}. The node v_{max} is the node that satisfies all the constraints of **h** for the quasi-identifiers, at the topmost level; in other words, v_{max} is the highest possible node that can obtain an exact answer to the user's request. We will also refer to v_{max} as the *highest conforming candidate* node. Then, two cases can hold: (a) v_{max} is able to provide an exact solution (Lines 4 - 13), or (b) it is not, and thus we have to resort to approximate suggestions to the user (Lines 14 – 20). The check on whether a node can provide an exact solution is given by function <u>checkExactSolution</u> that looks up the histogram of a node v and performs the appropriate check depending on the privacy criterion (k-anonymity, l-diversity, ...) . Note that this is the only part of the algorithm that needs to be customized according to the privacy criterion.

When the former case is concerned and an exact answer can be provided by the highest conforming candidate node v_{max}, then we can be sure that the sublattice induced by v_{max} contains an exact answer; however, we need to discover the one with the minimum possible height and, therefore, we need to descend down the lattice to discover it. For the case where the lowest possible height that contains a node that can return an answer that respects the constraints set by the user, we resolve the tie by choosing the node with the least suppression. The auxiliary variable *Candidates* holds all the nodes that conform to the user request, organized per height. Each time such a node is found, it is added to *Candidates* at the appropriate level (Line 5) and its descendants (returned via the function *lower*()) are recursively explored via the call of function ExactSublatticeSearch. When the lattice is appropriately explored we need to find lowest level with a solution in the lattice (Line 10) and, among the (several possible) solutions of this level we must pick the one with the least suppression (Line 11).

```
ExactSublatticeSearch(v,L,k,h,MaxSupp,Candidates){
        if v is visited then exit;
        mark v as visited;
        if (checkExactSolution(v,L,k,h,MaxSupp) == true){
                Candidates[height(v)] = Candidates[height(v)] U {v};
                for all v_c in lower(v)
                        ExactSublatticeSearch(v_c,L,k,h,MaxSupp,Candidates);
        }
}
```

Figure 4.4 Function Exact Sub lattice Search

```
checkExactSolution(v,L,k,h,MaxSupp){
        lookup histogram of v in L;
        if suppressed(v,k) <= MaxSupp && height(v) <= h  return true;
        else return false;
}
```

Figure 4.5 Function check Exact Solution

If the highest candidate node v_{max} fails to provide an answer that conforms to the user request, then we are certain that it is impossible to derive such a conforming answer from our lattice and we need to search for approximations. So, we provide the users with suggestions on the possible relaxations that can be made to his criteria. In this context, three suggestions are considered:

The first suggestion, provided by the invocation of function ApproximateMaxSupp, retains the privacy criterion k and the max tolerable height h fixed and tries to find the best possible solution with respect to the number of suppressed tuples. Since h is to be respected, again we are restricted in the sub-lattice induced by v_{max}. Since v_{max} has failed to provide a conforming answer, no node in the sublattice can provide such an answer, either. So, we assess the number of tuples that have to be suppressed if we retain k fixed and stay at the highest candidate node v_{max}. Observe that any node in the sublattice of v_{max} will result in higher or equal number of suppressed tuples (see the next section for a proof) – remember that the lower we go, the smaller the groups are and the higher the suppression. In other words, it will either be v_{max} that will give the answer or one of its descendants in the rare case that the groups of the descendant are mapped one to one to the groups of v_{max} thus resulting in exactly the same number of suppressed tuples.

The third suggestion is quite similar to the first: this time, function ApproximateK retains the height constraints **h** (again) and the maximum tolerable number of suppressed tuples *MaxSupp* and tries to determine what is the highest k that can provide this approximation. Again, for the same reasons as in the case of the approximation of suppression, we restrict our search to v_{max} (or any of its descendants that has a 1:1 mapping of groups to the ones of v_{max}).

```
ApproximateMaxSupp(L,v,k,h,MaxSupp){
        find the minimum amount of suppressed tuples, approxSupp, s.t.
checkExactSolution(v,L,k,h,approxSupp) returns true;
        if no such value exists, return {};
        else{
              for all v_c in sublattice(v) (recursively){
                    checkExactSolution(v_c,L,k,h,approxSupp)
                      break when a whole level fails to produce a solution;
              }
              let v_win be the node with the lowest height that satisfies k,h,approxSupp
              (with arbitrary tie resolution)
        return v_win,k,h,approxSupp;
        }
}
```

Figure 4.6 Function ApproximateMaxSupp

```
ApproximateK(L,v,k,h,MaxSupp){
        find the maximum value of k, approxK, s.t. checkExactSolution(v,L,approxK,h,maxSupp)
returns true;
        if no such value exists, return {};
        else{
             for all v_c in sublattice(v) (recursively){
                checkExactSolution(v_c,L,approxK,h,maxSupp)
                      break when a whole level fails to produce a solution;
             }
             let v_win be the node with the lowest height that satisfies approxK,h,maxSupp
                                                         (with arbitrary tie resolution)
            return v_win,approxK,h,maxSupp;
        }
}
```

Figure 4.7 Function ApproximateK

Finally, the second suggestion, provided by function <u>ApproximateH</u> retains the maximum tolerable number of suppressed tuples *MaxSupp* and the privacy criterion of *k* and tries to determine what is the lowest height h that can provide an answer for these constraints. This time, we operate outside the borders of the sublattice of $v_{\max}$ since **h** is not to be respected. The function <u>ApproximateH</u> performs a binary search on the height between the height of $v_{\max}$ and the upper possible height (the top of the lattice). Every time a level is chosen, we start to check its nodes for possible solutions via the function <u>checkIfNoSolutionInCurrentHeight</u>. If the function explores a height fully and fails to find an answer, this is an indication that we should not search lower than this height (remember: failure to find a solution signals for ascending in the lattice). Every time the function finds a node that can answer, then we must search in the lower heights for possibly lower solutions. At the end, the binary search stops and the value *currentMinHeight* signifies the lowest possible height where a solution is found. Then, we explore this height fully to determine the node with the minimum suppression.

```
ApproximateH(L,v,h_low,h_high,k,h,MaxSupp){
   while(h_low <= h_high){
         h_current = middle between h_low and h_high;
         flag = checkIfNoSolutionInCurrentHeight(L,h_current,k,MaxSupp);
         if (flag == true){
             low = current + 1;
         }
         else{
             currentMinHeight = current;
             high = current - 1;
         }
   }
   for all v_c in currentMinHeight, find the one v_win, with the minimum suppressed(v_c,k);
         //exception: this fails only if k > |R|, else top of the lattice always answers
         return v_win,k,height(v_win),MaxSupp;
}


checkIfNoSolutionInCurrentHeight(L,h_current,k,MaxSupp){
         for all v_c in h_current
         if suppressed(v_c,k) <= MaxSupp return false;
         return true;
}
```

Figure 4.8 Function Approximate H

4.2. Theoretical guarantees on the correctness of the proposed algorithm

In this subsection, we will discuss properties of the histogram-annotated lattice of generalization schemes and prove that our algorithm is correct.

Notation. We will employ the following notation:

lower(*v*) the set of nodes *u* who are connected to node *v* via a node (*u*,*v*) –i.e., the nodes whose generalization scheme is equal to *v*'s, with the exception of exactly one dimension where *u* is one level lower than *v*.

desc(*v*) the set of nodes *u* for whom a path exists towards *v*

L(*v*) the sublattice induced by a node *v* (i.e., the subset of the lattice whose nodes are either *v*, or, descendants of *v*)

cumKA(*v*|*k*) the number of suppressed tuples (y-value of the cumulative histogram) for node *v* when the cut-off constraint for k-anonymity (x-value of the cumulative histogram) is *k*.

cumSLD(*v*|*l*) the number of suppressed tuples (y-value of the cumulative histogram) for node *v* when the cut-off constraint for l-diversity (x-value of the cumulative histogram) is *l*.

Theorem 1. Assume a constraint on the height of hierarchies **h**=[h_1, ..., h_n]. Assume also the node $v_{\max}$ = [h_1, ..., h_n]. All the nodes of the full lattice that respect **h** are within the sub-lattice induced by $v_{\max}$ and there is no node outside the lattice induced by $v_{\max}$ that respects **h**.

Proof. Since $v_{\max}$ is the top element of the lattice, all nodes of the lattice have dimension heights lower or equal to the dimension heights of $v_{\max}$. Consequently, all nodes of the lattice induced by $v_{\max}$ respect **h** by definition. For a node *u* not to belong in the lattice, there must be at least one dimension whose height is higher than the respective height of $v_{\max}$. Then *u* does not respect the constraint of **h**. QED

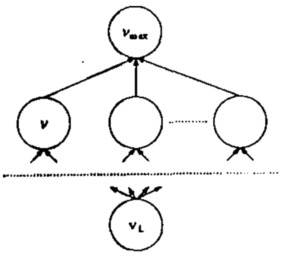
Given a node $v_{\max}$ that induces a sub-lattice, the groups of $v_{\max}$ are produced by aggregating the groups of its descendants in the sub-lattice. Then, for any value α the cumulative histogram for $v_{\max}$ has a smaller or equal value than the cumulative histogram for any node *v* in the descendants of $v_{\max}$. This holds both for k-anonymity and l-diversity. Formally:

Theorem 2. Given a node $v_{\max}$ and an integer any value α, the following hold:

$$cumKA(v_{\max}|\alpha) \leq cumKA(v|\alpha),\ v \in desc(v_{\max})$$

$$cumSLD(v_{\max}|\alpha) \leq cumSLD(v|\alpha),\ v \in desc(v_{\max})$$

Proof. This is almost direct consequence of the Rollup-property introduced in [LeDR05]. Assume the situation depicted in the following figure. Let $L(v_{\max})$ be the lattice between $v_{\max}$ as the top element and $v_\perp$ as the lowest element. Assume node *v* has a generalization scheme [l_1, l_2, ..., l_{n-1},l^*_n] while $v_{\max}$ has a scheme [l_1, l_2, ..., l_{n-1},l_n] and l_n = l^*_n + 1 (without loss of generality, we can assume that $v_{\max}$ differs from *v* only by one level in one dimension, whereas all the other dimensions are exactly the same; this practically works as the minimum possible distance between the two nodes).



Then (Rollup-property), there exists a *N*:1 mapping of values of l^*_n over l_n, *f*: *dom*(l^*_n)

$\rightarrow dom(l_n)$. f is a total function. A 1:1 mapping is also acceptable as a rare, special case in this setting. As a N:1 function, we know that for every l^*_n there exists exactly one value in l_n (but not obligatorily the inverse).

So, for every group $x^*[x_1, x_2, \ldots, x_{n-1}, x_n^*, count^*]$ that appears in v, there exists a group $x[x_1, x_2, \ldots, x_{n-1}, x_n, count]$ in v_{max}, and due to f, many groups of v (and, at least one) are potentially mapped to groups of v_{max} (but not vice versa). Therefore, for every such pair x, x^*, such that $f(x^*) = x$, $x^*.count^* \leq x.count$. Similarly, the same holds for the number of distinct sensitive values.

Remember now that the cumulative histogram records the tuples that are to be suppressed whenever a constraint on the minimum group size is given. There are three cases that concern us here:

(i) $\alpha \leq x^*.count^* \leq x.count$: in this case, neither x^* nor x would be suppressed with a request for α.

(ii) $x^*.count^* \leq x.count < \alpha$: both x^* and x would be suppressed with a request for α -- i.e., both would be counted in $cumKA(v|\alpha)$ and $cumKA(v_{max}|\alpha)$, respectively.

(iii) $x^*.count^* < \alpha \leq x.count$: in this case, x^* would be counted in $cumKA(v|\alpha)$ and x would not be counted in $cumKA(v_{max}|\alpha)$

For all these cases, it is impossible that a group is counted in $cumKA(v_{max}|\alpha)$ and its respective groups are not counted in the appropriate $cumKA(v|\alpha)$, $v \in desc(v_{max})$. On the other hand, unless a 1:1 mapping exists, there are groups counted in $cumKA(v|\alpha)$ but not in $cumKA(v_{max}|\alpha)$.

Exactly the same holds for *cumSLD*. QED

Corollary 2.1. Assume a user request $q = [k, \mathbf{h}, maxSupp]$ over a lattice $\mathcal{L}$ annotated with the cumulative histograms for a data set D. If all the nodes at height h violate q then it is impossible to find a node v at a height lower or equal than h that respects q.

Proof. Obvious.

Observation. Observe that the simple histograms, e.g., $KA(v|k)$ demonstrate arbitrary relationships for nodes and their descendants. For example, assume the following table T with a QI={*date*, *item*}, and the corresponding histograms at two different levels of the date dimension. The table has three sections. In the first section at the left, we depict 4 rows and their quasi-identifier values. The second section of the table in the middle, contains the counts per group at the {*day*, *item*} level. The third section, at the right, contains the counts per group at the {*month*, *item*} level.

Microdata		
1/Jan/2010	Cola	...
1/Jan/2010	Cola	...
2/Jan/2010	Milk	...
3/Jan/2010	Cola	...

Counts per QI group at day level		
QI value		*Count()*
1/Jan/2010	Cola	2
2/Jan/2010	Milk	1
3/Jan/2010	Cola	1

Counts per QI group at month level		
QI value		*Count()*
Jan/2010	Cola	3
Jan/2010	Milk	1

Here are also the histograms for the nodes {*day*,*item*} and {*month*, *item*}:

	k=1	*k=2*	*k=3*
Day	2	1	0
Month	1	0	1

Observe that for value *k*=2, the histogram of the lower-level node has a higher value than the histogram of the higher-level node. This is typical for small values of *k* which appear in the histograms of lower level nodes but disappear at higher levels, since the small groups of the lower level are merged in large groups of the higher level, resulting in the absence of small sized groups at the high level. At the same time, for value *k*=3 the opposite phenomenon is observed. Therefore, it is not possible to derive any theoretical guarantees for the simple histograms.

The following set of theorems guarantees that the proposed algorithm is correct. First we prove that once a node provides a solution (i.e., respects the three criteria posed by the user), we need to search its descendants for the lowest possible node that returns an answer, too.

Then, we deal with the case where the top-acceptable node fails to meet the user constraints and thus, we need to search for approximations.

Theorem 3. Assume a user request q = [k,**h**,*maxSupp*] over a lattice $\mathcal{L}$ annotated with the cumulative histograms for a data set *D*. Assume the top-acceptable node $v_{\max}$ that has **h** as its generalization scheme. If $v_{\max}$ respects q, then the node with the lowest height that respects q is in $L(v_{\max})$.

Proof. Obvious, due to Theorem 1 and Theorem 2: All the nodes that respect q are obligatorily in $L(v_{max})$ and there is at least one solution to the user request (the one of v_{max}). Theorem 2 does not disqualify the possibility that a node with lower height than v_{max} respects the constraints of q; therefore, we need to search for the best possible answer in $L(v_{max})$. QED.

Once the exact answering is covered, we need to consider the cases of the relaxations and answer the question: where should we search for possible relaxations if an exact answer is not there? So, assume the case where the criteria set by the user for **h** highlight node v_{max} which is unable to fulfill all three conditions and, we decide that the first approximation we want to explore involves relaxing *k*, respecting –at the same time- *maxSupp* and **h**.

Since we want to respect **h**, we must search for solutions within the lattice induced by v_{max}. Assume that v_{max} violates *k*, **h**, *maxSupp*. To relax the privacy criterion we need to find a smaller value than *k* which will have the property that *maxSupp* will be respected. Of course, we want to give the maximum possible privacy, so we need to find the maximum possible such value. We will use the notation k_r (standing for "relaxed k"), $k_r < k$, for the largest value that respects k_r, **h**, *MaxSupp* within the node v_{max}. However, there is a catch in the situation: it is not always possible to find such a value k_r. A clear (and actually, frequent at small heights) example for this situation is when the number of groups of size 1 at v_{max} is larger than *maxSupp*. Then, it is impossible to find a lower *k* that respects *maxSupp*.

In Theorem 4 we will show that, if a solution exists, then, in any of the nodes of the sublattice induced by v_{max} there is no value k^* which is larger than k_r and suppresses the same amount of tuples – in other words, it provides better k-anonymity with the sacrifice of the same amount of tuples. In Theorem 5 we will deal with the case where no solution can be found anyway.

Theorem 4. Assume a user request $q = [k,\mathbf{h},maxSupp]$ over a lattice $\mathcal{L}$ annotated with the cumulative histograms for a data set *D*. Assume that the top-acceptable node v_{max} which has **h** as its generalization scheme fails to respect *q*. Assume the largest value

k_r, $k_r < k$, such that v_{max} respects q_r=[k_r,**h**,*maxSupp*]. Then, there is no node v, $v \neq v_{max}$, $v \in L(v_{max})$, such that q^*=[k^*,**h**,*maxSupp*], $k^* > k_r$, is respected at v.

Proof. If v_{max} fails to meet q, then no node in $L(v_{max})$ can respect q. This holds for an query q_i=[k_r,**h**,*maxSupp*], $k_i < k$, too. If this did not hold, and there existed a node v, $v \neq v_{max}$, $v \in L(v_{max})$ such that k_i, **h**, *MaxSupp* is respected at v and then, $cumKA(v|k_i) < cumKA(v_{max}|k_i)$. Absurd by Theorem 2. QED.

Observation. What this theorem says is that the maximum possible value that we can get for the approximation of k is k_r. So, should we take v_{max} as the node that gives the solution? Practically, the answer is positive; however, theoretically, we need to perform an extra test. Observe that it is possible to have a situation where there is a 1:1 mapping between the groups of the higher level node v_{max} and the lower level node v (i.e., for every group of the ancestor node there is exactly one group of the descendant node). In this case, their histogram is exactly the same and v is a better solution than v_{max} (due to its lower height). This means that the descendants of v_{max} must be recursively searched for this possibility when we want to relax k. However, the search can be made in a breadth-first way; if a certain level does not have a node with the property of the 1:1 mapping, no further search should be performed. Moreover, this is a property that can be known offline, in advance.

Theorem 5. Assume a user request q = [k,**h**,*maxSupp*] over a lattice $\mathcal{L}$ annotated with the cumulative histograms for a data set D. Assume that the top-acceptable node v_{max} which has **h** as its generalization scheme fails to respect q. Assume there is no value k_r, $k_r < k$, such that v_{max} respects q_r=[k_r,**h**,*maxSupp*]. Then, there is no node v, $v \neq v_{max}$, $v \in L(v_{max})$, such that q^*=[k^*,**h**,*maxSupp*], $k^* > k_r$, is respected at v, for any value k^*.

Proof. If there is no value k_r at v_{max} that respects q, this means that the cumulative histogram at v_{max}, at position k_r, has already too many tuples to be suppressed. In other words, $cumKA(v_{max}|k_r) > maxSupp$. However, $cumKA(v_{max}|k_r)$ is the smallest amount of tuples to be suppressed for all $v \in L(v_{max})$. Consequently, if v_{max} fails to provide any value k_r that respects q^*, then no other node in $L(v_{max})$ can. QED

The easiest case is the case when we want to relax the amount of suppressed tuples.

Observation. Observe that for any node v, there is always a y-value (i.e., a number of suppressed tuples) for a fixed k^* at the cumulative histogram; this can be 0 if we are lucky and all groups are of size larger than k^*, or $|D|$ if we are unlucky and k^* is larger that the larger possible k the node can sustain.

Based on the above observation, when we deal with relaxing the suppression, there is only one issue, specifically, which is the node that for a fixed k, h will produce the minimum suppression. Not surprisingly, it turns out that this node is either $v_{\max}$ or one if its descendants that has a 1:1 mapping of groups with $v_{\max}$.

Theorem 6. Assume a user request $q = [k,\mathbf{h},maxSupp]$ over a lattice $\mathcal{L}$ annotated with the cumulative histograms for a data set D. Assume that the top-acceptable node $v_{\max}$ which has **h** as its generalization scheme fails to respect q. Assume the smallest value M, $M > maxSupp$, such that $v_{\max}$ respects $q_r=[k,\mathbf{h},M]$; actually, this is $M=cumKA(v_{\max}|k)$. Then, there is no node v, $v \neq v_{\max}$, $v \in L(v_{\max})$, such that $q^*=[\mathrm{k},\mathbf{h},M^*]$, $M^* < M_r$, is respected at v.

Proof. Since $M=cumKA(v_{\max}|k)$, by Theorem 2 this is the smallest possible $cumKA(v|k)$ for any $v \in L(v_{\max})$. QED.

Observation. If $v_{\max}$ does not respect q, then there is no information we can exploit concerning the height relaxation. The lowest possible solution that respects both k and $maxSupp$, if such a solution exists, is outside $L(v_{\max})$, but it can be found in any other node, at any height. So, we must search the entire lattice for the relaxation of **h** except for $L(v_{\max})$.

Theorem 7. Assume a user request $q = [k,\mathbf{h},maxSupp]$ over a lattice $\mathcal{L}$ annotated with the cumulative histograms for a data set D. Assume that the top-acceptable node $v_{\max}$ which has **h** as its generalization scheme. Then, the following hold:

- If $v_{\max}$ respects q, then the lowest node that can answer q is in $L(v_{\max})$.
- If $v_{\max}$ does not respects q, then (i) the relaxation of k and the relaxation of $maxSupp$ are provided by $v_{\max}$; (ii) we must search the entire lattice for the relaxation of **h**.

Proof. Directly from the above.

Observation. Based on all the above, the algorithm Simple Anonymity Negotiation is correct.

4.3. Experimental method

Goals. Our experiments are oriented towards assessing the following properties.

- *Effectiveness*. Given an initial request by the user with thresholds on the maximum tolerable amount of suppressed tuples, the maximum level of generalization per quasi-identifier attribute and the minimum acceptable value for the privacy criterion (either k for k-anonymity, or, *l* for simple l-diversity), how likely is it to obtain a completely acceptable solution for a given setup of data set, quasi-identifier set and sensitive attribute? The set of experiments aim to discover the effect of all the problem parameters to the likelihood of achieving an acceptable solution as opposed to the probability of needing to resort to an answer that relaxes one of the above constraints. We diagrammatically depict answers to queries with successful answer with light color and answers to queries that needed relaxation with blue (dark) color.
- *Efficiency*. Given the full lattice that is derived from the hierarchies of the quasi-identifier attributes and the full histogram for the privacy criterion under consideration, how fast can we obtain an answer to the user's request (either fully compliant with the user criteria, or a relaxed one, if this is not possible)? To assess the efficiency of the method, in every experiment we measure (a) the *number of visited nodes* of the lattice and (b) the total *execution time* needed to produce an answer (in msec). In all occasions, the reported execution times are the average of 5 executions of the same request of (lowest-acceptable-k, maxSupp, topmost node = constraint on all dimensions for the top tolerable level). Naturally, the number of visited nodes is always the same and the varying quantity is the time needed to retrieve an answer or a set of 3 possible relaxations.

Data sets. The data sets we have used are: (a) the Adult – Income data set from the UCI repository [UCI], (b) the IPUMS - data set downloaded from [IPUMS].

Parameters. We have tested algorithm Simple anonymity Negotiation for its efficiency and effectiveness over different data sets, quasi-identifier sizes, values for the privacy criterion for both k-anonymity and l-diversity, maximum allowed suppression levels and maximum allowed generalization heights per quasi-identifier attribute. For each data set and privacy criterion, we employ different values for these parameters, thus, we refer the reader to the subsequent subsections for more details on specific values.

Implementation specific data structures and database schema

In our implementation we retain the lattice in a database at the hard disk as a relation *Node* and a relation *Edges*. The relation edges are straightforward: $Edges(\underline{Start,\ End})$. The relation Nodes varies with the size of quasi-identifier set, as we retain two attributes per quasi-identifier dimension: (a) an attribute dim_i with the name of the dimension and an attribute $index_i$ with the level of the dimension that the node has. So relation *Nodes* is as follows: $Nodes(\underline{id},dim_1,index_1,.....,dim_n,index_n)$. A value [25,*age*, 2, *race*, 1, *work_class*,1] indicates the node with level 2 for age, level 1 for race and level 1 for work_class.

At the same time, in main memory we implement the following data structures:

- For each node of the lattice, we retain two lists that hold the histogram: the first list keeps the number of tuples and the second list keeps the number of tuples that pertain to every value of the histogram (i.e., position *i* in the list refers to value *I for* the x-axis of the histogram)
- We retain a collection of nodes that practically holds all the information for the nodes in main memory. In other words, we keep the id, the levels and the abovementioned histogram for every member of the collection that represents a node of the lattice. Also, we use an attribute to mark nodes as visited or not. We opted for a hash-based dictionary implementation of the collection, with id being the hash-value, in order to allow efficient lookup by id (remember that the edges hold id's of nodes, so whenever we move up or down from a node this implementation comes handy).

Of course, apart from the above, we also keep the data sets in the appropriate databases. In the following, we list the database schemata for each of the two data sets we have employed. We depict the sensitive attribute in teletype letters.

<u>Database schema for the Adult data set</u>

Adult(Id,Age,Gender,Race,Marital_Status,Native_country,Work_Class,Occupation,Salary,`Hours per week`*)*

Age (level$_0$*, level*$_1$*, level*$_2$*, level*$_3$*, level*$_4$*)*

Race(level$_0$*, level*$_1$*, level*$_2$*)*

Marital_status(level$_0$*, level*$_1$*, level*$_2$*, level*$_3$*)*

Education(level$_0$*, level*$_1$*, level*$_2$*, level*$_3$*, level*$_4$*)*

Occupation(level$_0$*, level*$_1$*, level*$_2$*)*

Work Class(level$_0$*, level*$_1$*, level*$_2$*, level*$_3$*)*

<u>Database schema for the Ipums dataset</u>

Adult(id, age, education, birthplace, gender, `occupation`*)*

Age (level$_0$*, level*$_1$*, level*$_2$*, level*$_3$*, level*$_4$*)*

Education(level$_0$*, level*$_1$*, level*$_2$*, level*$_3$*, level*$_4$*)*

Birthplace(level$_0$*, level*$_1$*, level*$_2$*, level*$_3$*)*

Gender(level$_0$*, level*$_1$*)*

Configuration. In all our experiments we have used a Core Duo 2.5GHz server with 3GB of memory and 300GB (7200 RPM) hard disk. The operating system was Ubuntu 8.10 and the database server was MySQL 5.0.67. The code is written in Java in Eclipse IDE.

4.4. Findings for k-anonymity over the Adult data set

In this subsection, we discuss our experimental findings when working with the Adult data set and k-anonymity as our privacy criterion.

Table 4.3 Experimental parameters and possible values

<table>
<tr><th></th><th>|QI|=3</th><th>|QI|=4</th><th>|QI|=5</th><th>|QI|=6</th></tr>
<tr><td>Generalization level constraints</td><td>101, 211 (default), 212</td><td>1001, 2011 (default), 2112</td><td>11001, 21012 (default), 22112</td><td>111001, 211012 (default), 222112</td></tr>
<tr><td></td><td colspan="4">For all QI's, we have used three configurations: (a) a low one, with all levels constrained low in their hierarchies, (b) a middle-low (default) with some constraints placed on levels in the middle of their hierarchies and (c) middle, with all levels constrained at the middle in their hierarchies</td></tr>
<tr><td>k</td><td colspan="4">3, 10 (default), 50</td></tr>
<tr><td>MaxSupp</td><td colspan="4">32, 321 (default), 3216 (approx. 0.1%, 1%, 10% of the data set)</td></tr>
</table>

4.4.1. Effect of k over time costs

In this sequence of experiments we modify the value of minimum tolerable k and assess its impact to the number of visited nodes and execution time. All experiments operate with a fixed set of values for the rest of the parameters, and specifically:

- Maximum allowed number of suppressed tuples = 321
- The constraint on the uppermost tolerable level is 211, 2011, 21012, 212012 for |QI|=3, ..., 6 – i.e., in every dimension, we place a constraint approximately up to the middle of its hierarchy.

When |QI| is small (|QI|=3), the maximum number of suppressed tuples pushes the solution lower than the starting level (which is the maximum tolerable level). So, the algorithm recursively descends towards 0,0,..,0. As k increases, the solution is found earlier.

In all other cases, the cost in terms of visited nodes increases sub-linearly with k.

There is a single exception to the sublinear increase, and this is the case of k = 50 and QI = 6, where the number of visited nodes drops. This is due to the fact that the binary search over the height was successful quick enough and gave a quick correct answer (you can see an example of such a binary search in Figure 4.9).

Parameters: 10 32 211012	**Parameters:**10 321 211012	**Parameters:**10 3216 211012
Start: low:7 high:18 **Current:** 12 Find, and check 34 nodes **Current:** 9 Don't find, check 496 nodes **Current:** 10 Don't find, check 469 nodes **Current:** 11 Find, and check 286 nodes **Solution at level** :11 Check 396 nodes	**Start:** low: 7 high: 18 **Current:** 12 Find, and check 2 nodes **Current:** 9 Find, and check 416 nodes **Current:** 7 Don't find, check 396 nodes **Current:** 8 Don't find, check 469 nodes **Solution at level** :9 Check 495 nodes	**Start:** low: 7 high: 18 **Current:** 12 Find, and check 1 nodes **Current:** 9 Find, and check 4 nodes **Current:** 7 Find, and check 116 nodes **Solution at level** :7 Check 395 nodes

Figure 4.9 Example of binary search for Variant max supp (QI-6) that detects a solution early enough

The important observation here is that the number of nodes visited increases dramatically with |QI| with a scale factor of 5 (approximately) for every extra attribute added to the QI set (i.e., the values of QI=5 are 5 times greater than the respective values of QI = 4; the same approximately happens when we increase QI to 6). In terms of time, the experiments do not take more than 3,5 msec for QI=4,5. QI = 3 takes longer (between 5 and 1 msec) due to the recursive call to search in lower levels of the hierarchy. QI = 6 makes between 6 and 8 msec for all three values of k.

There are certain cases, where the relaxation for the decreasing of k does not return a result. These cases do not induce a significant overhead, since the search is locally performed in the topmost node.

4.4.2. Effect of height constraints over time costs

In this sequence of experiments we modify the constraints over the maximum tolerable generalization heights per attribute and assess the impact of these heights to the number of visited nodes and execution time. All experiments operate with a fixed set of values for the rest of the parameters, and specifically:

- Lowest tolerable k = 10
- Maximum allowed number of suppressed tuples = 321

We employ three variants for the topmost node, for each QI. The three variants involve constraints in all the dimensions of the QI set (and thus, a respective a topmost node) in the following 3 fashions:

(a) every dimension is constrained by a level that is low in the hierarchy,

(b) some dimensions are constrained low in the hierarchy and some are constrained in the middle

(c) all dimensions are constrained in the middle of their hierarchy

Specifically, the constraints employed are as follows:

Table 4.4 Constrains for the Experiment

	Low	Low-middle	Middle
QI=3	101	211	212
4	1001	2011	2112
5	11001	21012	22112
6	111001	211012	222112

The findings for the effect of the constraints in the hierarchy are as follows:

- The lower the constraint is, the more search for finding adequate relaxation is required. In other words, when the constraints are set low, it is impossible to obtain an answer at the topmost acceptable node and thus, we need to climb a lot in the lattice until we reach a tolerable relaxed solution. On the contrary, when the constraint is in the middle, the required climbing is less.
- The time required for the operation to complete is typically analogous to the number of visited nodes. All experiments for QI = 4 and 5 run between 2 and 6 msec. The case of QI = 6 induces an extra overhead with times between 5 and 8 msec.
- An exception to all the above findings is the case where the QI is small (|QI| = 3). In this case, the topmost node is adequate for an answer and the algorithm recursively descends towards the best possible answer. The times needed are in the range of 1 and 6 msec.
- Another observation here is that when the topmost node is low in the lattice, it is quite frequent that the relaxation of k (keeping the topmost node and the maxSupp fixed) fails. This is due to the fact that when we are low in the lattice, the suppressed tuples are too many and possibly even k=2 is not

sufficient to fulfill the requirements for maxSupp. On the other hand, when the topmost node is relatively higher in the lattice, the number of suppressed tuples is lower; thus, finding a relaxed answer is feasible.

Again, the important finding is that, ultimately, the dominating factor in terms of time and number of visited nodes is the size of the QI. The rule of the scale factor of 5 for every extra attribute in the QI seems to be preserved (see Figure 4.10; the shaded areas in Fig. 4.10 depict cases where the search was directed downwards in the sublattice of v_{max} and a drop in the values is observed as QI increases; the cells not defined are cases where we move from a downwards search to an upwards search).

Height	**3→4**	**4→5**	**5→6**
Low	-	5,91	4,75
Low-middle	-	5,78	6,20
Middle	0,90	6,00	3,31

Figure 4.10 Scale up in number of visited nodes as QI size increases for different values of the height constraint

4.4.3. Effect of maxSupp over time costs

In this sequence of experiments we modify the value of maximum tolerable amount of suppressed tuples and assess its impact to the number of visited nodes and execution time. All experiments operate with a fixed set of values for the rest of the parameters, and specifically:

- Lowest tolerable k = 10
- The constraint on the uppermost tolerable level is 211, 2011, 21012, 212012 for |QI|=3, ..., 6 – i.e., in every dimension, we place a constraint approximately up to the middle of its hierarchy.

The maximum allowed number of suppressed tuples takes the following values: 32, 321, 3216.

Typically, the number of visited nodes logarithmically decreases as the value of maxSupp increases (each time by a factor of 10). This is clearly due to the fact that the higher the number of tolerable number of suppressed tuples is, the easier it is to find a solution. In fact, when the experiments operate on the largest possible value of maxSupp (i.e., 3216 suppressed tuples), all QI's expect for the case of QI = 6 achieve an acceptable solution in the topmost node and move downwards to obtain a better solution. The times needed for the cases of QI = 4, 5, 6 range between 1 and 8 msecs. There are exceptions to the above general observation, which we list:

- The case of QI=3 has the peculiarity that the topmost node achieves an acceptable solution for a maxSupp of 321. In this case, we observe that the higher the maxSupp, the more time it takes to find a good solution, since too many nodes qualify for acceptable solutions. In all cases, the recursive search for a better solution is much costlier than the search for an approximate solution of maxSupp = 32 in QI = 3. The time ranges between 2 msecs for the approximate search and 5msecs for the costliest exact search.

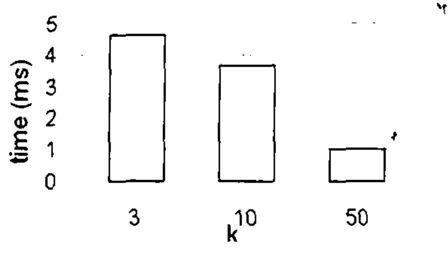
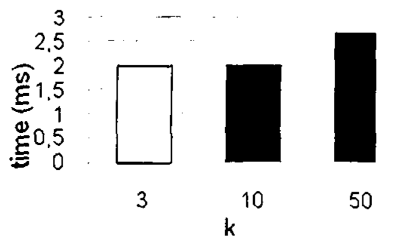
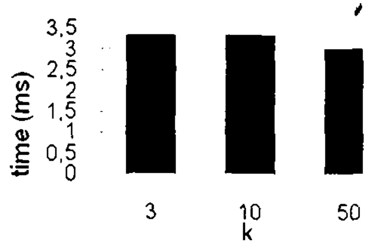
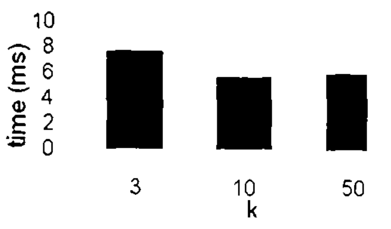
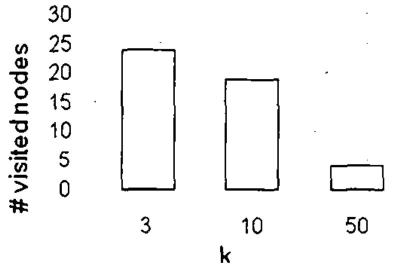
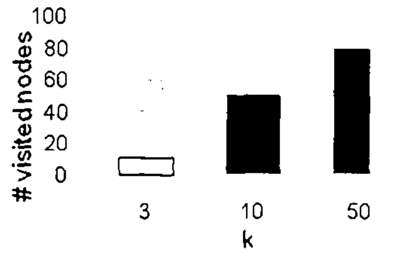
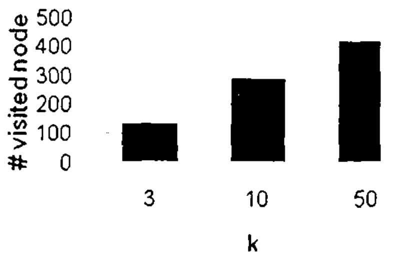
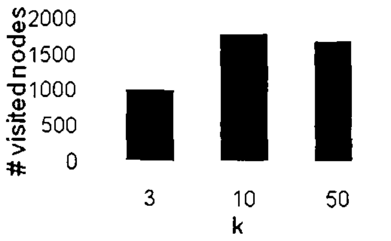
- The case of maxSupp =321 and QI = 6 breaks the general rule, as it is costlier than the case of maxSupp = 32 in terms of visited nodes.

Again, the dominating factor in terms of cost is the size of the QI. The rule of scale-up in terms of 5 is broken: the increase for every extra attribute in the QI results in approximately 3 to 6 times more visited nodes. In Figure 4.11, we can observe this scaling up on the upper left part of the figure. Also, the shaded areas in Figure 4.11 depict cases where the search was directed downwards in the sublattice of $v_{\max}$ (where a drop in the values is observed as QI increases) and the cells not defined are cases where we move from a downwards search to an upwards search.

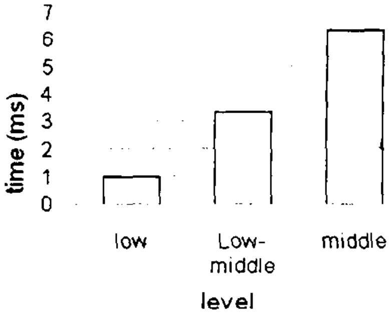
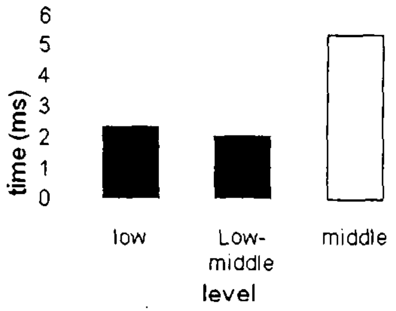
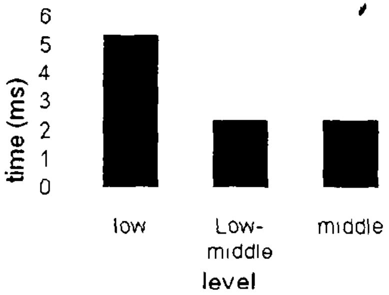
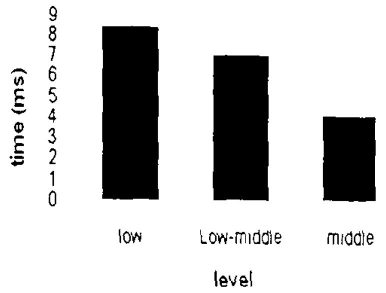
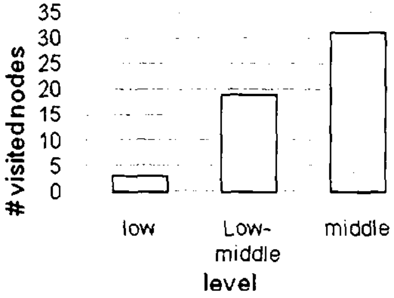
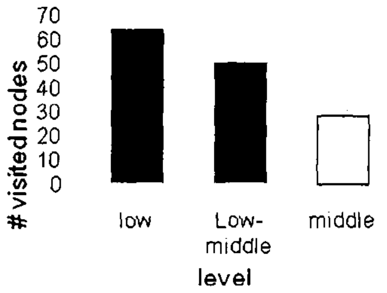
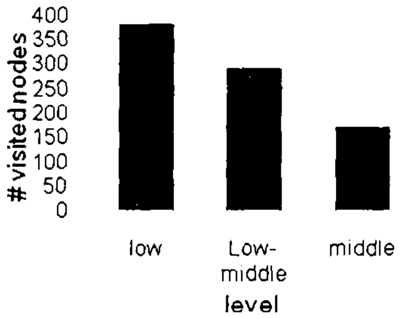
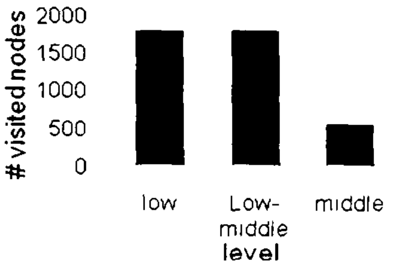
maxSupp	3→4	4→5	5→6
32	3,20	6,10	3,47
321	-	5,78	6,20
3216	0,86	0,26	-

Figure 4.11 Scale up in number of visited nodes as QI size increases for different values of maxSupp

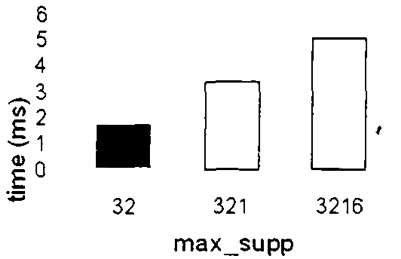
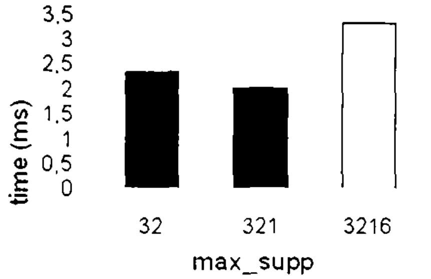
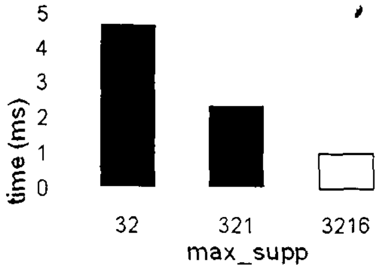
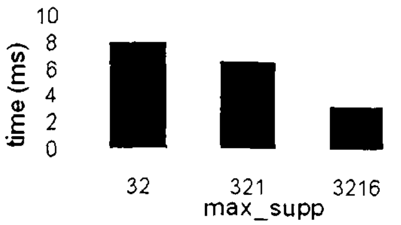
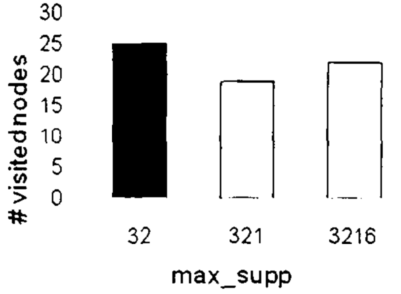
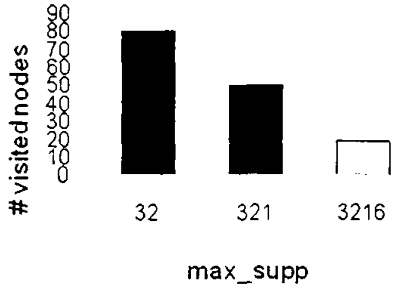
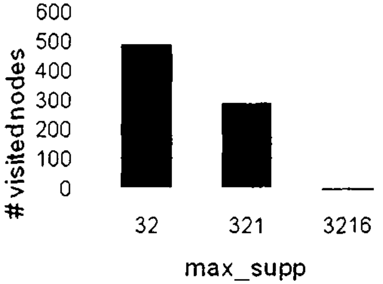
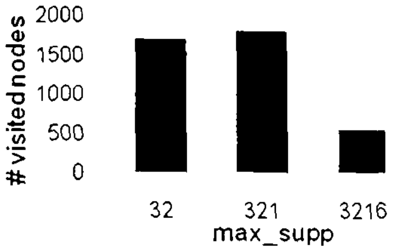
Variant k

|QI|=3	|QI|=4	|QI|=5	|QI|=6
			
			
Parameters: 3, 321, 211 ***Solution:*** Move down find, id: 4 supp tuples: 125 level:100 **Parameters:** 10,321,211 ***Solution:*** Move down find, id: 5 supp tuples:170 level:110 **Parameters:** 50, 321, 211 ***Solution:*** Move down find, id: 23 supp tuples:251 level:211	**Parameters:** 3, 321, 2011 ***Solution:*** Move down find, id: 56 supp tuples:283 level:1011 **Parameters:** 10, 321, 2011 ***Solution:*** Rlx1: id:65 supp tuples:655 level:2011 Rlx2 id:17 supp tuples:170 level:1210 Rlx3 id:65 k:5 Supp_tupp:301 **Parameters:** 50, 321, 2011 ***Solution:*** Rlx1: id:65 supp tuples:4077 level:2011 Rlx2 id:107 supp tuples:137 level:1212 Rlx3 id:65 k:5 Supp tpls:301	**Parameters:** 3, 321, 21012 ***Solution:*** Rlx1: id:551 supp tuples:656 lv:21012 Rlx2 id:503 supp tuples:108 lv:10212 Rlx3:No solution. **Parameters:**: 10, 321, 21012 ***Solution:*** Rlx1: id:551 supp tuples:2533 lvl:21012 Rlx2: id:504 supp tuples:230 lvl:10222 Rlx3: No solution. **Parameters:** 50, 321, 21012 ***Solution:*** Rlx1: id:551 supp tuples:9214 lvl:21012 Rlx2 id:636 supp tuples:169 lvl:40122 Relaxation3 : No solution.	**Parameters:** 3, 321, 211012 ***Solution:*** Rlx1: id:2591 supp tpls:1611 lvl:211012 Rlx2 id:763 supp tpls:155 lvl:400202 Rlx3: No solution. **Parameters:** 10, 321, 211012 ***Solution:*** Rlx1: id:2591 supp tpls:5362 lvl:211012 Rlx2 id:2171 supp tpls:285 lvl:401211 Rlx3: No solution. **Parameters:** 50, 321, 2 1 1 0 1 2 ***Solution:*** Rlx1 id:2591 supp tpls:15106 lv:211012 Rlx2 id:2812 supp tpls:137 lvl:401222 Rlx3: No solution.

Variant constraint on upper levels

|QI|=3	|QI|=4	|QI|=5	|QI|=6
			
			
Parameters: 10, 321, 101 **Solution:** Move down find, id: 19 supp tuples:257 level:101 **Parameters:** 10, 321, 211 **Solution:** Move down find, id: 5 supp tuples:170 level:110 **Parameters:** 10, 321, 212 **Solution:** Move down find, id: 5 supp tuples:170 level:110	**Parameters:** 10, 321, 1001 **Solution:** Rlx1: id:55 supp tuples:2349 lvl:1001 Rlx2 id:17 supp tuples:170 lvl:1210 Rlx3: No solution **Parameters::** 10, 321, 2011 **Solution:** Rlx1: id:65 supp tuples:655 lvl:2011 Rlx2 id:17 supp tuples:170 lvl:1210 Rlx3 id:65 k:5 Supp_tupp:301 **Parameters:** 10, 321, 2112 **Solution:** Move down find, id: 59 supp tuples:285 level:1111	**Parameters:** 10, 321, 11001, **Solution** Rlx1: id:280 supp tpls:8169 lvl:11001 Rlx2 id:504 supp tuples:230 lvl:10222 Rlx3: No solution. **Parameters::** 10, 321, 21012 ***Solution:*** Rlx1: id:551 supp tuples:2533 lvl:21012 Rlx2: id:504 supp tuples:230 lvl:10222 Rlx3: No solution. **Parameters:** 10, 321, 22112 **Solution:** Rlx1: id:563 supp tpls:369 lvl:22112 Rlx2 id:635 supp tpls:60 lvl:40112 Rlx3 id:563 k:9 Supp tpls:315	**Parameters:** 10, 321, 111001 **Solution:** Rlx1: id:336 supp tpls:12823 lvl:111001 Rlx2 id:2171 supp tpls:285 lvl:401211 Rlx3: No solution. **Parameters:** 10, 321, 211012 **Solution:** Rlx1: id:2591 supp tpls:5362 lvl:211012 Rlx2 id:2171 supp tpls:285 lvl:401211 Rlx3: No solution. **Parameters:** 10,321, 222112 **Solution:** Rlx1: id:2623 supp tpls:712 lvl:222112 Rlx2 id:2811 supp tpls:54 lvl:401212 Rlx3 id:2623 k:5 Supp tpls:298

Variant max_supp

|QI|=3	|QI|=4	|QI|=5	|QI|=6
			
			
Parameters: 10, 32, 211**Solution:** Rlx1: id:23 supp tuples:55 level:211 Rlx2 id:11 supp tuples:28 level:310 Rlx3 id:23 k:7 Supp_tupp:31 **Parameters:** 10, 321, 211**Solution:** Move down find, id: 5 supp tuples:170 level:1 1 0 **Parameters:**: 10, 3216, 211**Solution:** Move down find, id: 1 supp tuples:1921 level:000	**Parameters:** 10, 32, 2011 **Solution:** Rlx1: id:65 supp tuples:655 lvl:2011 Rlx2 id:35 supp tuples:28 lvl:3210 Rlx3 No solution **Parameters:** 10, 321, 2011 **Solution:** Rlx1: id:65 supp tuples:655 lvl:2011 Rlx2: id:17 supp tuples:170 lvl:1210 Rlx3: id:65 k:5 Supp_tupp:301 **Parameters:** 10, 3216, 2011 **Solution:** Move down find, id: 19 supp tuples:2110 level:2000	**Parameters:** 10, 32, 21012 **Solution:** Rlx1: id:551 supp tpls:2533 lvl:21012 Rlx2: id:638 supp tpls:14 lvl:40212 Rlx3: No solution **Parameters:** 10, 321, 21012 **Solution:** Rlx1: id:551 supp tpls:2533 lvl:21012 Rlx2: id:504 supp tpls:230 lvl:10222 Rlx3: No solution **Parameters:** 10, 3216, 21012 **Solution:** Move down find, id: 551 supp tuples:2533 level:21012	**Parameters:** 10, 32, 211012 **Solution:** Rlx1: id:2591 supp tpls:5362 lvl:211012 Rlx2: id:2812 supp tpls:21 lvl:401222 Rlx3: No solution **Parameters:** 10, 321, 211012 **Solution:** Rlx1: id:2591 supp tpls:5362 lvl:211012 Rlx2: id:2171 supp tpls:285 lvl:401211 Rlx3: No solution **Parameters:** 10, 3216, 211012 **Solution:** Rlx1: id:2591 supp tpls:5362 lvl:211012 Rlx2: id:1525 supp tpls:1222 lvl:400210 Rlx3: id:2591 k:5 Supp tpls:2915

4.5. Findings for l-diversity over the Adult data set

In this subsection, we discuss our experimental findings when working with the Adult data set and l-diversity as our privacy criterion. The assessed measures and parameters are the same with the ones discussed in our experimental method section and section 4.3 (for k-anonymity over the Adult data set). The only difference, of course, concerns the values we have used for l, which are summarized as follows:

Table 4.5 Parameters of the Algorithm and experiments for l-diversity

Offline	**Used value**
Lattice	Full lattice construction
<X-value, Y-value>	groupSize for l-anonymity, counter
On-line	
Privacy constraint	l-diversity
QoS()	Height of a solution
Experiments	
L	3, 6 (default), 9

4.5.1. Effect of l over time costs

In this sequence of experiments we modify the value of minimum tolerable *l* and assess its impact to the number of visited nodes and execution time. All experiments operate with a fixed set of values for the rest of the parameters, and specifically:

- Maximum allowed number of suppressed tuples = 321
- The constraint on the uppermost tolerable level is 211, 2011, 21012, 212012 for |QI|=3, .., 6 – i.e., in every dimension, we place a constraint approximately up to the middle of its hierarchy.

As with the case of k-anonymity, we can observe that as *l* increases, the cost scales up very slowly with the increase of l. One might blame the choice of the values for *l* (i.e., we could have picked significantly larger values for l), but this is not correct: a value of l=9 at the bottom level introduces a suppression of 15% (for QI = 3) to 93% (for QI = 6) at the bottom node of the lattice and 0.6% (for QI=3) to 38% for 11…1 (for QI=6). The times are always very small and range between 1 and 8 ms. The times reported are subject to phenomena of DBMS caching and thus are not in complete accordance to the numbers of visited nodes; however, the fluctuations in the time needed for the execution of the algorithm are unimportant and due to the few IO's incurred in our implementation as well as DBMS caching. Remember that the lattice is small and in general, it can fit in main memory: for QI = 6 we have 3600 nodes and 15960 edges. Also remember that in our experiments, for the sake or program simplicity, we keep the edges in a relation at the hard disk, while we keep the nodes with their histograms in main memory.

At the same time, the cost increases dramatically with the increase of QI.

Observe also that for a small QI (and small l) – i.e., for the cases with small numbers of suppressed tuples-- the possibility of finding an acceptable solution within the constraints expressed by the user is significant. So, for QI=3 as well as for QI=4 and l=3, the algorithm found an acceptable solution at the topmost node of the user's constraint and recursively climbed down the lattice to find a better solution. For the rest of the cases, the algorithm produced approximate solutions; interestingly for high values of QI and l, the algorithm could not provide an approximation for a lesser value of *l* (depicted as Rlx3 in the detailed results).

4.5.2. Effect of height constraints over time costs

In this sequence of experiments we modify the constraints on the maximum possible height for the quasi-identifier attributes and assess the impact of the height vector to the number of visited nodes and execution time. All experiments operate with a fixed set of values for the rest of the parameters, and specifically:

All experiments operate with:

- Lowest tolerable l= 6
- Maximum allowed number of suppressed tuples = 321

Again, as in the case of k-anonymity we fix three combinations of values for the height constraints: (a) all quasi-identifiers are constrained low in their hierarchy; (b) some quasi-identifiers are constrained low and some in the middle of the hierarchies and (c) all quasi-identifiers are constrained in the middle of their hierarchies. The specific values for these constraints are the ones reported in section 4.3.2.

The findings for the effect of the constraints in the hierarchy are practically the same with k-anonymity and can be summarized as follows:

- Exactly as in the case of k-anonymity, the lower the constraint is, the more search (both in terms of number of nodes visited and time spent) for finding adequate relaxation is required, since the solution that satisfies both *l* and maxSupp is found further up in the lattice. When the QI size is small (QI=3 or 4), it is possible that middle and low-middle constraints result in exact answers (which, in turns, are produced by a recursive descent down the lattice from the topmost node of the specified constraints).
- All the times needed to provide the user with an exact or approximate answer fall between 2 and 6 msec. Interestingly, the number of nodes visited and the required times are also very similar to the ones of k-anonymity.
- Again, as in the case of k-anonymity, the relaxation of *l* frequently fails to deliver a solution.
- Again, as in the case of k-anonymity, the size of the QI is the dominating factor for the cost; every extra attribute in the QI incurs a scale up of 3 – 5 in terms of both visited nodes and time spent.

4.5.3. Effect of maxSupp over time costs

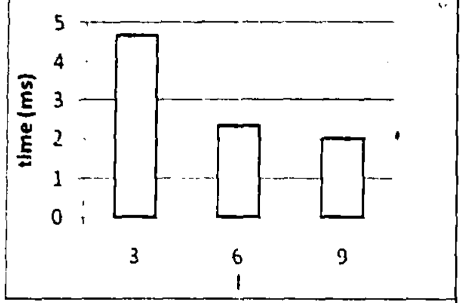
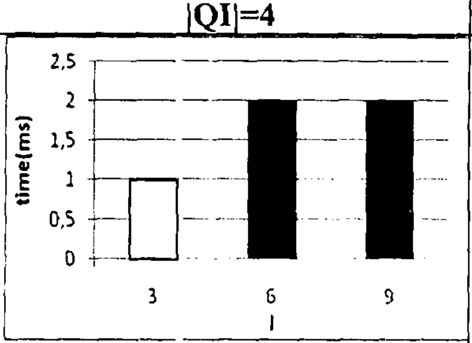
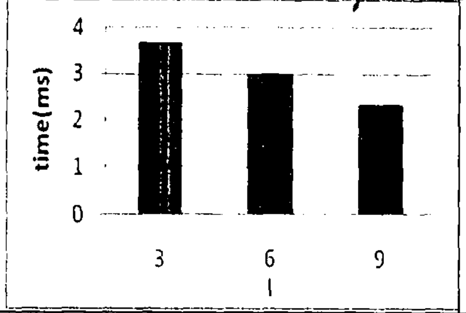
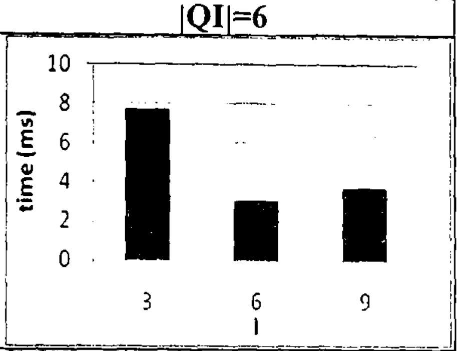
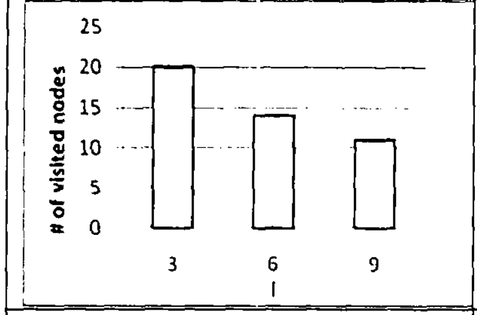
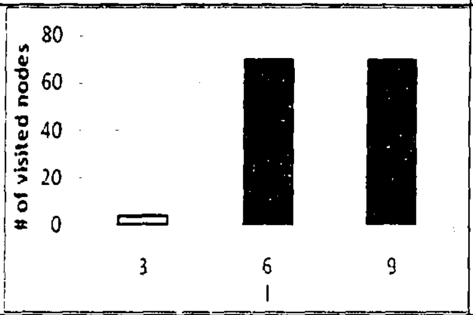
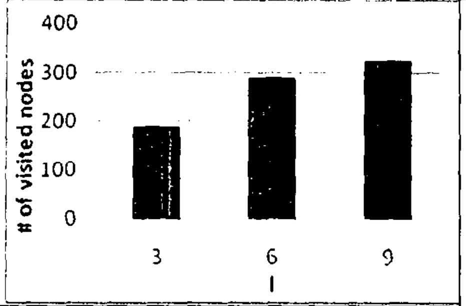
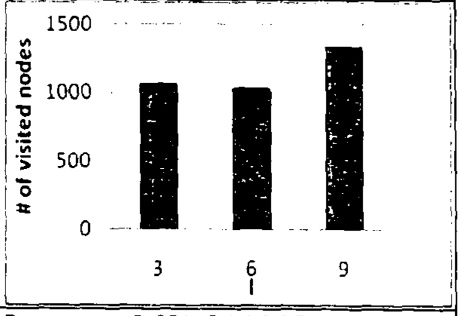
In this sequence of experiments we modify the constraint on the maximum possible amount of suppressed tuples and assess its impact to the number of visited nodes and execution time. All experiments operate with a fixed set of values for the rest of the parameters, and specifically:

- The maximum tolerable amount of suppressed tuples takes the values: 32, 321, 3216
- Lowest tolerable *l* = 6
- The constraint on the uppermost tolerable level is 211, 2011, 21012, 212012 for |QI|=3, .., 6 – i.e., in every dimension, we place a constraint approximately up to the middle of its hierarchy

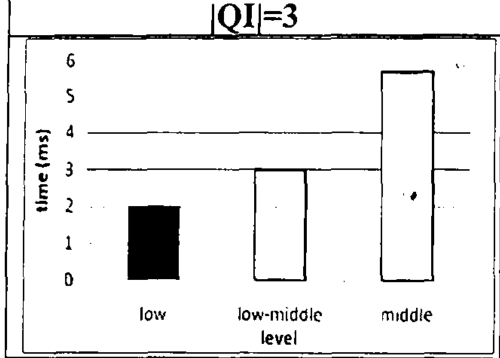
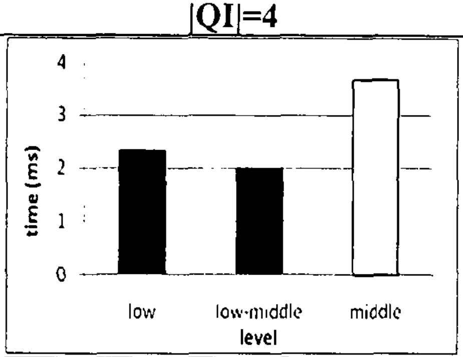
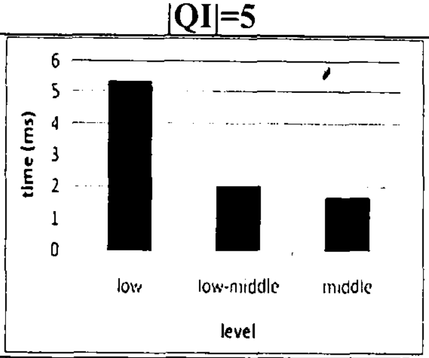
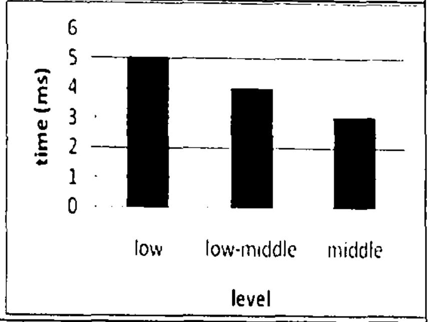
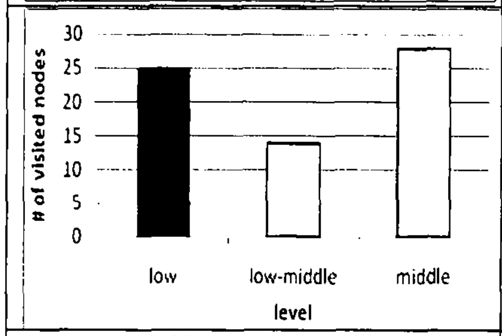
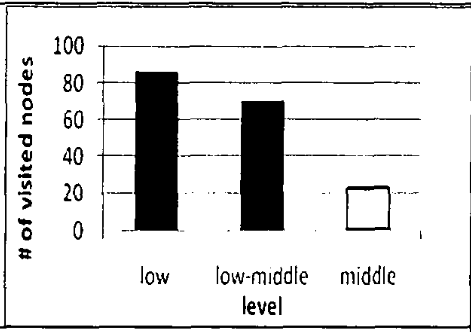
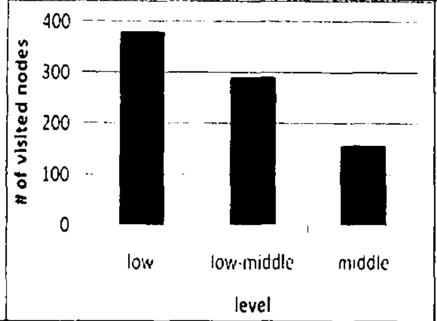
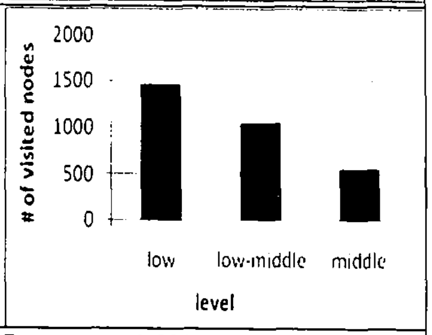
As in the case of the other experiments, here too the results are remarkably similar to the ones of k-anonymity.

- The higher the maximum tolerable number of suppressed tuples is, the faster we get a solution when the result is approximate. This is due to the fact that the answer is found lower in the lattice. The opposite holds when the answer is exact (e.g., in the case of QI=3, where it is possible to attain an exact answer); in this case, the first possible answer is attained at the topmost node and then the algorithm descends to find the best possible answer, resulting in higher execution times.
- The costs in terms of time and visited nodes are quite similar to ones of k-anonymity. The time costs range between 1 and 7 msec.
- The size of the QI is once again the dominant factor and each extra attribute in the quasi-identifier set incurs a scale up of 4-5 times more visited nodes.

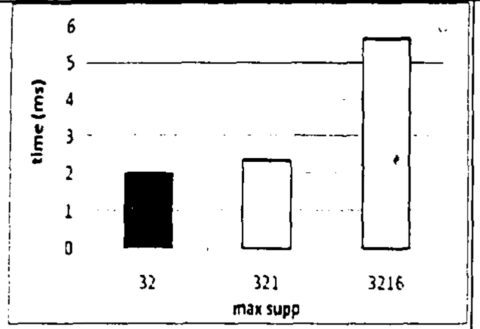
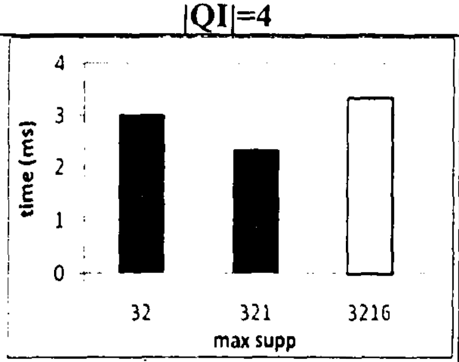
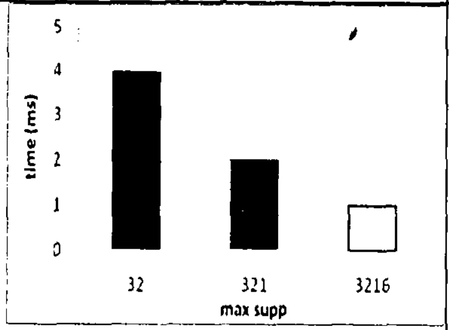
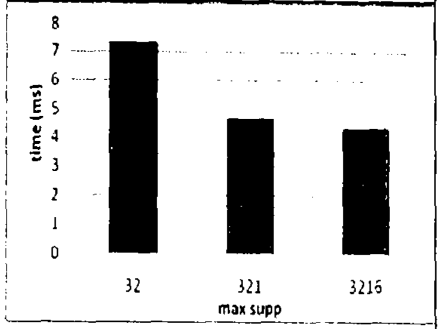
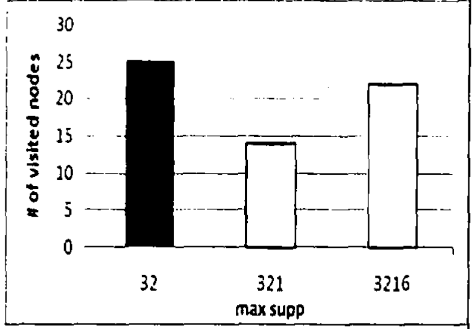
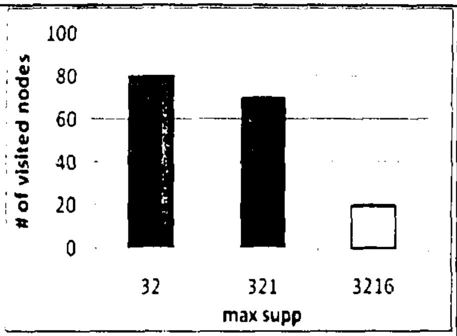
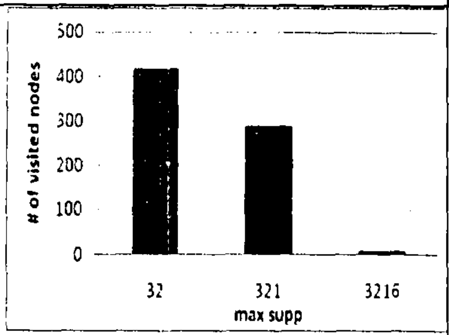
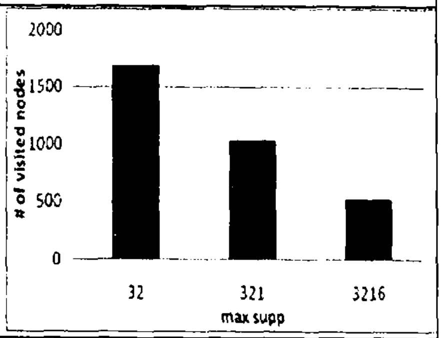
l-diversity over Adult. Variant value for the privacy criterion, l

|QI|=3	|QI|=4	|QI|=5	|QI|=6
			
			
Parameters:3, 321, 2 1 1 ***Solution:*** Move down find, id: 4 supp tpls:240 lvl:100 **Parameters**:6, 321, 2 1 1 ***Solution:*** Move down find, id: 20 supp tpls:70 lvl:111 **Parameters**:9, 321, 2 1 1 ***Solution:*** Move down find, id: 20 supp tpls:186 lvl:111	**Parameters**:3, 321, 2 0 1 1 ***Solution:*** Move down find, id: 65 supp tpls:319 lvl:2011 **Parameters**:6, 321, 2 0 1 1 ***Solution:*** Rlx1: id:65 supp tpls:1013 lvl:2011 Rlx2 id:18 supp tpls:54 lvl:1220 Rlx3 id:65 l:3 Supp tpls:319 **Parameters**:9, 321, 2 0 1 1 ***Solution:*** Rlx1: id:65 supp tpls:2005 lvl:2 0 1 1 Rlx2 id:18 supp tpls:104 lvl:1 2 2 0 Rlx3 id:65 l:3 Supp_tupp:319	**Parameters**:3, 321, 2 1 0 1 2 Rlx1: id:551 supp tpls:1139 lvl:2 1 0 1 2 Rlx2 id:503 supp tpls:244 lvl:1 0 2 1 2 Rlx3 No solution **Parameters**:6, 321, 2 1 0 1 2 ***Solution:*** Rlx1: id:551 supp tpls:3205 lvl:21012 Rlx2 id:504 supp tpls:302 lvl:10222 Rlx3 No solution **Parameters**:9, 321, 2 1 0 1 2 ***Solution:*** Rlx1: id:551 supp tpls:5418 lvl:21012 Rlx2 id:635 supp tpls:250 lvl:40112 Rlx: No solution	**Parameters**:3, 321, 2 1 1 0 1 2 ***Solution:*** Rlx1: id:2591 sup tpls:2715 lvl:211012 Rlx2 id:2452 sup tpls:288 lvl:101222 Rlx3 No solution **Parameters**:6, 321, 2 1 1 0 1 2 ***Solution:*** Rlx1: id:2591 sup tpls:6602 lvl:211012 Rlx2 id:2811 sup tpls:90 lvl:401212 Rlx3 No solution **Parameters**:9, 321, 2 1 1 0 1 2 ***Solution:*** Rlx1: id:2591sup tpls:10139 lvl:21101 2 Rlx2 id:2811 supp tpls:206 lvl:401212 Rlx3 No solution

l-diversity- Adult Variant constraint on upper levels

|QI|=3	|QI|=4	|QI|=5	|QI|=6
			
			
Parameters:6, 321, 1 0 1 **Solution:** Rlx1: id:19 supp tpls:368 lvl:1 0 1 Rlx2 id:6 supp tpls:54 lvl:1 2 0 Rlx3 id:19 l:5 Supp_tupp:266 **Parameters**:6, 321, 2 1 1 **Solution:** Move down find, id: 20 supp tpls:70 lvl:111 **Parameters**:6, 321, 2 1 2 **Solution:** Move down find, id: 34 supp tpls:64 lvl:102	**Parameters**:6, 321, 1 0 0 1 **Solution:** Rlx1: id:55 supp tpls:3081 lvl:1 0 0 1 Rlx2 id:18 supp tpls:54 lvl:1 2 2 0 Rlx3 No Solution **Parameters**:6, 321, 2 0 1 1 **Solution:** Rlx1: id:65 supp tpls:1013 lvl:2 0 1 1 Rlx2 id:18 supp tpls:54 lvl:1 2 2 0 Rlx3 id:65 l:3 Supp_tupp:319 **Parameters**:6, 321, 2 1 1 2 **Solution:** Move down find, id: 104 supp tpls:61 lvl:1112	**Parameters**:6, 321, 1 1 0 0 1 **Solution:** Rlx1: id:280 supp tpls:9444 lvl:11001 Rlx2 id:504 supp tpls:302 lvl:10222 Rlx3 No Solution **Parameters**:6, 321, 2 1 0 1 2 ***Solution:*** Rlx1: id:551 supp tpls:3205 lvl:21012 Rlx2 id:504 supp tpls:302 lvl:10222 Rlx3 No solution **Parameters**:6, 321, 2 2 1 1 2 **Solution:** Rlx1: id:563 supp tpls:434 lvl:22112 Rlx2 id:635 supp tpls:63 lvl:40112 Rlx3 id:563 l:5 Supp_tupp:282	**Parameters**:6, 321, 1 1 1 0 0 1 **Solution:** Rlx1: id:336sup tpl:14076 vl:111001 Rlx2 id:2811 supp tpls:90 lvl:401212 Rlx3 No Solution **Parameters**:6, 321, 2 1 1 0 1 2 **Solution:** Rlx1: id:2591suptpl:6602 lvl:211012 Rlx2 id:2811 supp tpls:90 lvl:401212 Rlx3 No Solution **Parameters**:6, 321, 2 2 2 1 1 2 **Solution:** Rlx1: id:2623 sup tpl:857 lvl:222112 Rlx2 id:2811 supp tpl:90 lvl:401212 Rlx3 id:2623 l:3 Supp_tupp:253

l-diversity- Adult Variant max supp

|QI|=3	|QI|=4	|QI|=5	|QI|=6
			
			
Parameters:6, 32, 2 1 1 **Solution**: Rlx1:id:23 supp tpls:57 lvl:2 1 1 Rlx2 id:35 supp tpls:22 lvl:1 1 2 Rlx3 id:23 k:4 Supp_tupp:25 **Parameters**:6, 321, 2 1 1 **Solution**: Move down find, id: 20 sup tpls:70 lvl:111 **Parameters**:6, 3216, 2 1 1 **Solution**: Move down find, id: 1 sup tls:2476 lvl:000	**Parameters**:6, 32, 2 0 1 1 **Solution**: Rlx1: id:65 supp tpls:1013 lvl:2 0 1 1 Rlx2 id:41 supp tpls:12 lvl:4 1 1 0 Rlx3 No Solution **Parameters**:6, 321, 2 0 1 1 **Solution**: Rlx1: id:65 supp tpls:1013 lvl:2 0 1 1 Rlx2 id:18 supp tpls:54 lvl:1 2 2 0 Rlx3 id:65 l:3 Supp_tupp:319 **Parameters**:6, 3216, 2 0 1 1 **Solution**: Move down find, id: 19 supp tpls:2826 lvl:2 0 0 0	**Parameters**:6, 32, 2 1 0 1 2 **Solution**: Rlx1: id:551 supp tpls:3205 lvl:21012 Rlx2 id:638 supp tpls:7 lvl:4 0 2 1 2 Rlx3 No Solution **Parameters**:6, 321, 2 1 0 1 2 **Solution**: Rlx1: id:551 supp tpls:3205 lvl:21012 Rlx2 id:504 supp tpls:302 lvl:10222 Rlx3 No Solution **Parameters**:6, 3216, 2 1 0 1 2 **Solution**: Move down find, id: 551 supp tpls:3205 lvl:21012	**Parameters**:6, 32, 2 1 1 0 1 2 **Solution**: Rlx1: id:2591 sup tpl:6602 lvl:211012 Rlx2 id:2823 supp tpls:7 lvl:403212 Rlx3 No Solution **Parameters**:6, 321, 2 1 1 0 1 2 **Solution**: Rlx1: id:2591 sup tpl:6602 lvl:211012 Rlx2 id:2811 supp tpls:90 lvl:401212 Rlx3 No Solution **Parameters**:6, 3216, 2 1 1 0 1 2 **Solution**: Rlx1: id:2591 sup tpl:6602 lvl:211012 Rlx2 id:503 sup tpl:1583 lvl:400201 Rlx3 id:2591 l:3 Supp_tupp:2715

4.6. Findings over the IPUMS data set

In this subsection, we discuss our experimental findings when working with the IPUMS data set and both k-anonymity and l-diversity as our privacy criterion. As already mentioned, the PUMS data set has a quasi-identifier size QI=4. The parameters we have used are as follows:

Table 4.6 Parameters for IPUMS experiments

k	3,30,50,100,150
l	3,6,10
Topmost node	low (1010), middle-low (2110), middle (2220)
MaxSupp	600, 6000, 60000

Unless otherwise stated when we vary a parameter, the rest of the parameters are pinned to the middle of the above values.

Our findings are qualitatively the same as with the case of the Adult data set when the QI is small both in terms of time and visited nodes. Here, we should point out again that: it is the lattice that matters and not the data size.

k-anonymity. As k increases there is a certain limit above which it is not possible to obtain exact answers. For small k's, where an exact answer is possible, the higher the value of k, the faster this solution is computed (remember: the algorithm is driven downwards the lattice recursively; a higher value of k stops the descent earlier). For higher values of k where we seek an approximation upwards in the lattice, the number of visited nodes increases as the need for a higher k drives the solution higher in the lattice.

As the height of the topmost acceptable node increases, the solution is found faster for the case of rather low topmost nodes (where approximate answers are required). When the topmost node is set in the middle of the lattice, the exact solution is found fast (the search is downwards the lattice and completes quickly). As the maximum tolerable number of suppressed values increases, the same phenomenon is also observed: the higher the value, the easier to obtain a solution – for large values of *maxSupp*, we can even attain an exact answer quite easily.

All the above results are very similar to the observations we have done for the Adult data set too.

l-diversity. The case of l-diversity demonstrates a completely different picture than k-anonymity. Apart from a couple of cases, all the other attempts for a solution lead to exact answers. First, let us state that things appear to operate as expected when exact answers are attained:

- As the value of *l* increases, the determination of a solution completes faster as a large value of *l* is prohibitive for several of the low-level solutions
- As the height of the topmost node increases, the solution is computed slower (as the beginning of the descent starts higher)
- As the maximum tolerable number of suppressed tuples increases, the solution is also slower since the search can go to higher depths in the lattice.

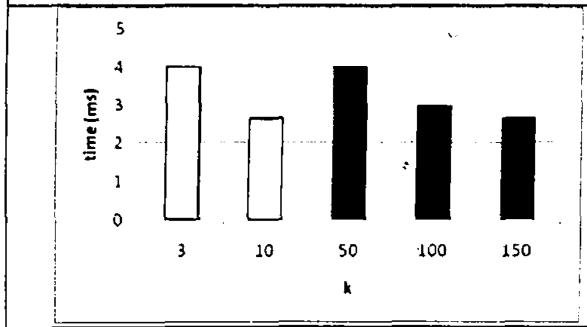
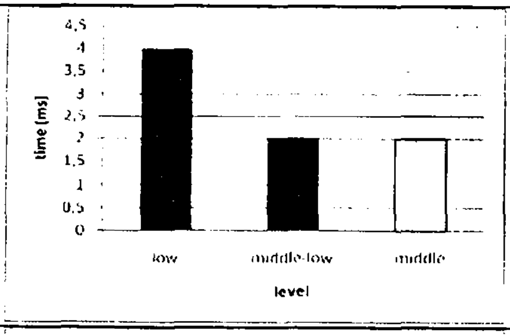
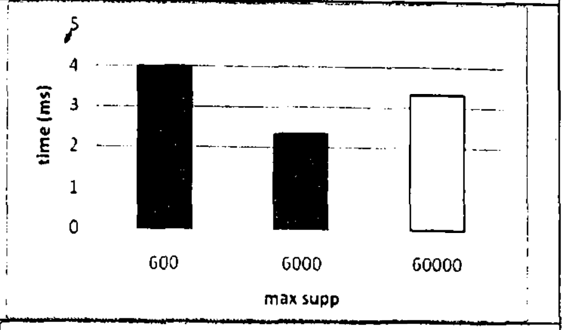
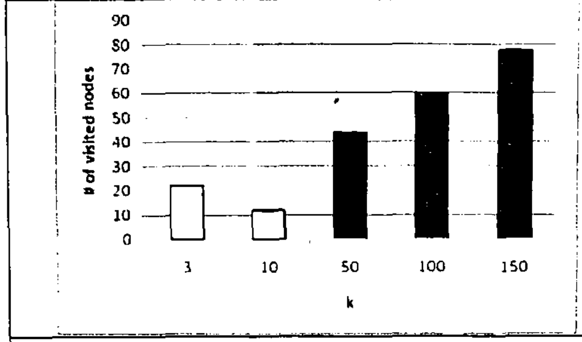
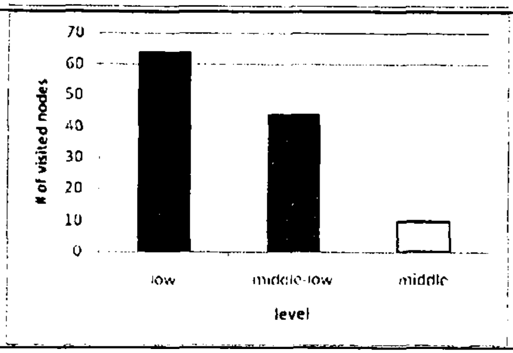
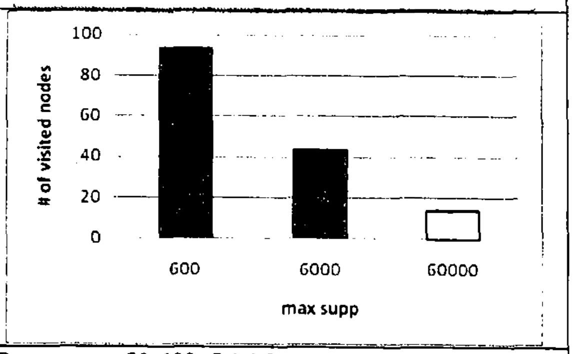
The interesting part is that the data set behaved quite close to the case of QI=3 of the Adult data set (and, thus, differently from the rest of the QI sizes of the Adult data set). Clearly this is due to the value of *l*: as the data size increases and the domains are comparable, the groups are larger and, most importantly, the possibilities for different sensitive values within a group are higher.

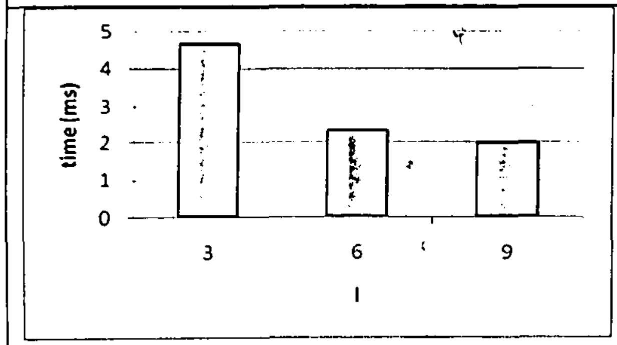
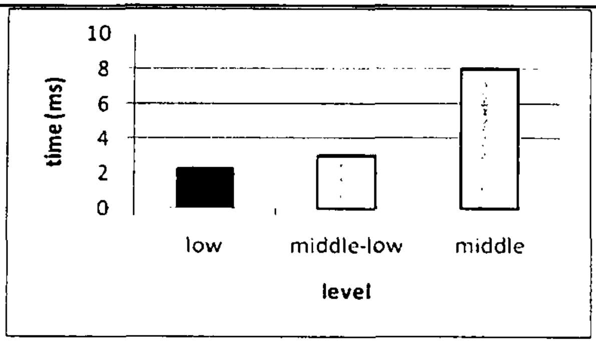
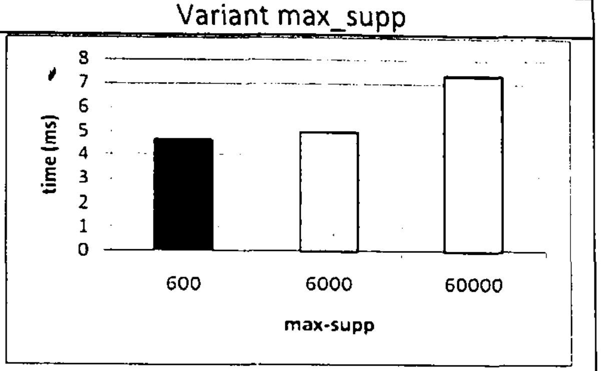
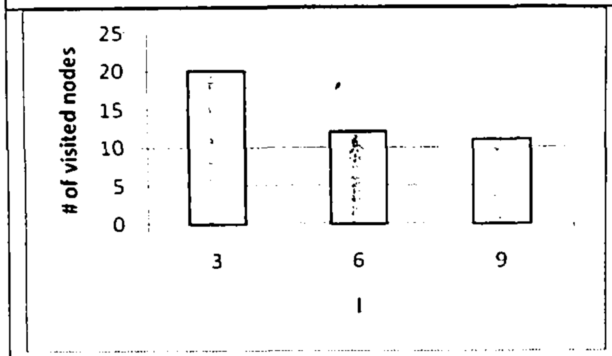
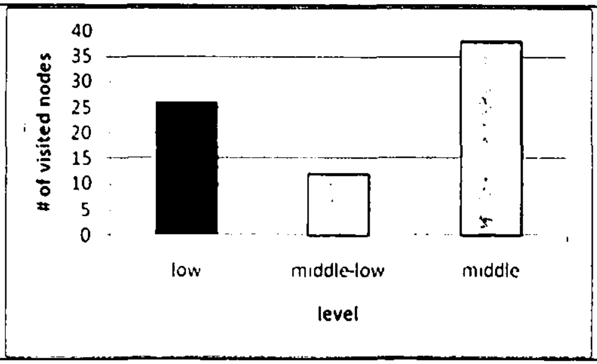
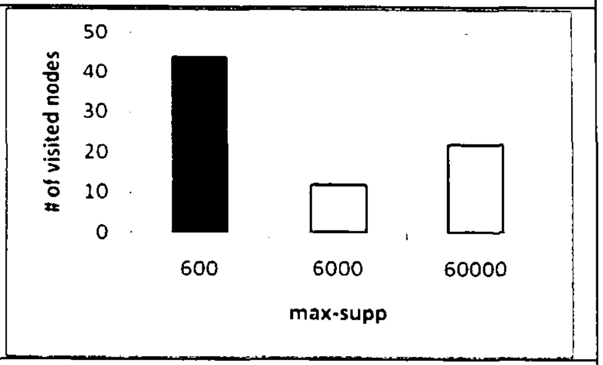
4.7. Summary of findings

We have experimented with algorithm Simple Anonymity Negotiation over two data sets and with two privacy criteria: k-anonymity and l-diversity. We have assessed the performance in terms of time and visited nodes as we vary the value for the privacy criterion, the maximum tolerable generalization height and the maximum tolerable amount of suppressed tuples. Our findings can be summarized as follows:

- The increase of the privacy criterion has divergent effects. When QI is small, there is an exact answer and the search is directed towards lower heights. Consequently, as k increases the solution is found earlier. On the contrary, for larger QI sizes and relaxations to user request, the increase of k sublinearly increases the search space.
- The increase of the maximum tolerable height has similar behavior. When the QI size is small, we can have exact solutions and the height increase increases the search space. In all other occasions where relaxations are sought, the higher the constraint, the faster a solution is found.

- The constraint on the maximum tolerable amount of suppressed tuples has also a similar behavior: the higher the constraint is set, the faster an approximate solution is found (except for low QI sizes where exact answers are possible and the behavior is inverse)
- In all experiments, it is clear that the costs are dominated by the QI size.
- Finally, in all experiments, the times ranged between 1 and 8 msec, thus facilitating the online, interactive negotiation of privacy with the user.
- The experiments with l-diversity demonstrate a similar behavior as the experiments of k-anonymity. Similarly, the IPUMS data set presents similar behavior as compared to the Adult data set. The only exception is the case of l-diversity, where the IPUMS offered exact answers quite frequently compared to their frequency in the case of the Adult data set; however, the behavior of the algorithm is identical to the one in the case of the Adult data set both for exact and approximate answers.

K-anonymity IPUMS (QI=4)		
Variant k	Variant level	Variant max_supp
		
		
Parametres:3, 6000, 2110 **Solution:** Move down find, id: 6 supp tuples:4462 lvl:0100 **Parametres:**10, 6000, 2 1 1 0 Move down find, id: 26 supp tpls:3778 lvl:1100 **Parametres:**50, 6000, 2 1 1 0 **Solution:** Rlx1: id:47 supp tuples:9476 level:2 1 1 0 Rlx2: id:36 supp tuples:2037 level:1 3 0 0 Rlx3: id:47 k:34 supp tpls:5948 **Parametres:**100, 6000, 2 1 1 0 **Solution:** Rlx1: id:47 supp tuples:19968 level:2 1 1 0 Rlx2: id:36 supp tuples:4775 level:1 3 0 0 Rlx3: id:47 k:34 Supp_tupp:5948 **Parametres:**150, 6000, 2 1 1 0 **Solution:** Rlx1: id:47 supp tuples:32572 level:2 1 1 0 Rlx2: id:86 supp tuples:2482 level:4 1 0 0 Rlx3: id:47 k:34 Supp_tupp:5948	**Parameters:**50, 6000, 1 0 1 0 **Solution:** Rlx1: id:22 supp tuples:110173 level:1 0 1 0 Rlx2 id:36 supp tuples:2037 level:1 3 0 0 Rlx3 id:22 k:3 Supp_tupp:4146 **Parameters:**50, 6000, 2 1 1 0 **Solution:** Rlx1: id:47 supp tuples:9476 level:2 1 1 0 Rlx2: id:36 supp tuples:2037 level:1 3 0 0 Rlx3: id:47 k:34 Supp_tupp:5948 **Parameters:**50, 6000, 2 2 2 0 **Solution:** Move down find, id: 51 supp tuples:4742 level:2 2 0 0	**Parameters:**50, 600, 2 1 1 0 **Solution:** Rlx1: id:47 supp tuples:9476 level:2 1 1 0 Rlx2: id:86 supp tuples:527 level:4 1 0 0 Rlx3: id:47 k:5 Supp_tupp:470 **Parameters:**50, 6000, 2 1 1 0 **Solution:** Rlx1: id:47 supp tuples:9476 level:2 1 1 0 Rlx2: id:36 supp tuples:2037 level:1 3 0 0 Rlx3: id:47 k:34 Supp_tupp:5948 **Parameters:**50, 60000, 2 1 1 0 **Solution:** Move down find, id: 26 supp tuples:30578 level:1 1 0 0

<table>
<tr><th colspan="3">l-diversity IPUMS (|QI|=4)</th></tr>
<tr><th>Variant l</th><th>Variant level</th><th>Variant max_supp</th></tr>
<tr>
<td> </td>
<td> </td>
<td> </td>
</tr>
<tr>
<td> </td>
<td> </td>
<td> </td>
</tr>
<tr>
<td><b>Parametres:</b>3, 6000, 2 1 1 0
<b>Solution:</b>
Move down find, id: 6 supp tpls:5333 level:0 1 0 0
<b>Parametres:</b>6, 6000, 2 1 1 0
<b>Solution:</b>
Move down find, id: 26 supp tpls:2779 level:1100
<b>Parametres:</b>9, 6000, 2 1 1 0
<b>Solution:</b>
Move down find, id: 46 supp tpls:2492 level:2 100</td>
<td><b>Parametres:</b>6, 6000, 1 0 1 0
<b>Solution:</b>
Rlx1: id:22 supp tpls:15973 level:1 0 1 0
Rlx2: id:26 supp tpls:2779 level:1 1 0 0
Rlx3: id:22 k:3 supp tpls:5151
<b>Parametres:</b>6, 6000, 2 1 1 0
<b>Solution:</b>
Move down find, id: 26 supp tpls:2779 level:1100
<b>Parametres:</b>6, 6000, 2 2 2 0
<b>Solution:</b>
Move down find, id: 26 supp tpls:2779 level:1100</td>
<td><b>Parametres:</b>6, 600, 2 1 1 0
<b>Solution:</b>
Rlxt1: id:47 supp tpls:1055 level:2 1 1 0
Rlx2: id:36 supp tpls:89 level:1 3 0 0
Rlx3: id:47 k:4 supp tpls:427
<b>Parametres:</b>6, 6000, 2 1 1 0
<b>Solution:</b>
Move down find, id: 26 supp tpls:2779 level:1100
<b>Parametres:</b>6, 60000, 2 1 1 0
<b>Solution:</b>
Move down find, id: 6 supp tpls:20837 level:0100</td>
</tr>
</table>

CHAPTER 5. PARTIAL LATTICE CONSTRUCTION

So far, we have seen that the on-line part of the determination of an exact or approximate anonymization scheme is completed within milliseconds for all possible combinations of quasi-identifier size, data size, privacy criterion and so on. This way is it is clear that the user can interact in real time with a negotiation system that (a) answers anonymization requests and (b) guides the user to different alternatives if the exact answer to his request is not feasible.

At the same time, the precomputation of the lattice at full scale, comprising all the histograms for every node of the lattice presents several problems as:

- It requires a non-negligible amount of space
- It requires a non-negligible amount of time to be computed
- It has to be fully recomputed when updates occur (unless auxiliary data structures are also kept)

- It scales up exponentially with the number of quasi identifier attributes and their hierarchies.

One could possibly argue that space is not really a problem in the sense that the lattice's size is dependent on the size of the underlying data, but on the size of the quasi-identifier set and the accompanying hierarchies. The computation of the histograms for each node, on the other hand, is an aspect that deserves attention. In our experiments, we have observed that the construction times for the full lattices annotated with histograms take up time in the order of half an hour for a quasi-identifier size of up to 6 and simple privacy criteria like k-anonymity and l-diversity. Clearly, this can be tolerated in certain applications, however, it is possible that some applications may not tolerate even this amount of time. To address this problem, in this section we explore different variants of the pre-processing step, where instead of generating the full lattice, we either opt to precompute a part of the lattice's histograms, or we generate the histogram of only the node that we visit in each move we make over the lattice. We refer to these alternatives as (a) **partial**, or, (b) **on-line computation of the lattice**. Of course, in these cases, we pay the price of not necessarily obtaining the optimal answer. In the rest of this section, we explore these alternatives and assess their impact on the effectiveness (quality of solution) and efficiency (time to build and explore the lattice) of the respective algorithms.

5.1. Partial lattice construction and the grouping power of generalization levels

The first possibility that we can explore is the *partial computation of the lattice which has to be based on carefully selecting which nodes to generate*. This is done on the basis of the effect that different attributes have to the relationship of suppression and privacy criterion. Overall, our method proceeds as follows:

1. An estimation of the effect that different levels have to suppression is made; on the basis of this estimation, each node in the lattice is also ranked with respect to the possible effect it can have to suppression.
2. The histograms for a specific percentage *p*% of the nodes of the lattice are computed.
3. Once this preprocessing is completed, the partial lattice is ready for usage; then, a modified version of the algorithms of section 4 is used to address user requests.

As we have already seen, not all dimensions and not all levels are equally affecting the amount of suppression we need to perform. Clearly, it is desirable that our node selection process picks lattice nodes that reduce the amount of suppression. Since we want to avoid generating all the lattice's histograms, we have to resort to prediction / estimation methods for the suppression power of a candidate node. To this end, a possible alternative to explore is the estimation of the importance of attributes to the suppression process. In the first of the following subsections we discuss the metrics we use for estimating the grouping power of *levels*; in the subsequent subsection we discuss how we exploit these metrics in order to achieve the desirable, i.e., the prediction of the grouping power (and thus, its effect to suppression) for *lattice nodes*.

Before proceeding we would like to remind the reader that relations involve identifier attributes that are removed, quasi-identifier attributes that are candidates for generalization, sensitive attributes that are to be protected and indifferent attributes that play no role in the generalization process. In general, we assume that a relation R is defined as $R(A_{\text{ID}}, A_1, \ldots, A_n, X_1, \ldots, X_m, S)$, where A_{ID} is an identifier, $A_1, \ldots, A_n$ is the quasi-identifier set, $X_1, \ldots, X_m$ are the indifferent attributes and S is the sensitive value.

5.2. The grouping power of hierarchy levels and its effect to suppression

Both the case of k-anonymity and the case of l-diversity suggest that the larger the groups are, the less suppression we need to perform. Therefore, it would be desirable to be able to identify levels that produce large groups and promote them against levels that do not have this property. We use two fundamental metrics, the first concerning the average group size produced by a level and the second concerning the importance of a level as compared to its previous level in the same hierarchy.

Average Group Size. We estimate the effect that a generalization to a level A^h will have to suppression via the average group size for this level. To compute the average group size we perform a simple query where (a) we group the relation R by the quasi-identifier set at the detailed level for all the quasi-identifiers but the one into investigation who is generalized to level h, and (b) we compute the average group size for this generalization scheme by diving the size of R with the number of groups that are formed in the previous step. Technically, if the quasi-identifier set is A_1, ..., A, ..., A_n, then the group by clause of the query that retrieves the number of groups is set to A_1, ..., A^h, ..., A_n. Bear in mind that this is quite different than simply grouping by A^h. The former produces the groups and their cardinality for all the combination of dimension A with the rest of the dimensions whereas the latter will only produce a number of tuples per value of the domain of A^h. This way we can also get reasonable estimations for the topmost level of a hierarchy, too (as opposed to the production of a single group that grouping by A^{All} would produce). Observe also the role of the indifferent attributes here: the primary key of the relation does not obligatorily comprise only quasi-identifiers; however our method works for any configuration with or without indifferent attributes (in the latter case, the average group size of the lowest level is 1).

Relative Importance of Generalization Levels. The average group size of a level is a quite powerful indicator of the effect a level has to the suppression; however it is not the only one. Whenever a certain dimension (e.g., the dimension *Age* as we shall see in the examples for the Adult data set) consistently produces large group sizes, it dominates the decisions on the possible generalizations that we should consider. It is possible, for example, that both levels Age^2 and Age^3 produce good large group sizes, but the benefit from moving from level 2 to level 3 in the age dimension might be small (i.e., Age^2 does a pretty good job, and despite the fact that Age^3 produces larger groups it would be better to generalize another dimension one level up; unfortunately, average group size does not give us this information). So, we need to introduce a metric that captures the relative importance of a level within its dimension – i.e., as compared to levels of the same dimension. To this end, we define the *relative importance of a generalization level* A^h, i.e., of an attribute in dimension *A* at height *h* as the fraction

$$relImp\,(A^h) = \begin{cases} \dfrac{avgGroupSize(A^h)}{avgGroupSize(A^{h-1})}, & \text{for all heights } h \text{ in } All, \ldots, 1, \text{ or} \\[2em] \dfrac{1}{power(A^1)}, & \text{for } h=0 \end{cases}$$

A first comment on the internals of the relative importance measure, observe the recursion in the definition: we can define the relative importance of a level as the increase in group size with respect to its lower level; of course, this is impossible for the lowest, most detailed level of a hierarchy. To define the relative importance of this lowest level, we could choose among alternative scores. We avoided the lowest possible score of 0, since setting the value to 0 would be unfair for nodes of the form 400000 (that would score much lower than they deserved). We also avoided setting the score to 1 (as one would normally expect) as a score of 1 is too close to several scores of middle-height levels (as we observed during our experiments) and this would result again in unfair rankings. Finally, we opted for fine-tuning the importance of the lowest level of each hierarchy to the inverse of the importance of level 1 as (a) it makes sense in terms of intuition and (b) it appears to work fine in practice. In terms of the introduced recursion, it is clear that we can always perform the computation of *relImp*(A^h) for all possible values of *h*.

Experimental findings. To illustrate the concept of the relative importance of each level, we list the measurements for the Adult data set. Remember that in our experiments we have used *Age, Work Class, Race, Occupation, Education and Marital Status* as the quasi-identifier, *Gender* and *NativeCountry* as indifferent attributes (the former because it only has two values and the second because it is too biased for the value USA; thus, they both tend to be generalized always) and *Hours per Week* as the sensitive attribute. In Table 5.1, we present both the relative importance of attributes organized per hierarchy as well as the total ordering of attributes by their importance.

The results are not surprising at all: as already observed in the previous experiments, the age hierarchy presents remarkable improvements when we chose to use it for generalization. This is due to the vast domain of its levels, compared to the other attributes (observe that *age*^{1} is the most strong attribute and the best choice to direct efforts for generalization that minimize suppression and, not surprisingly, *age*^{0} is the weakest attribute to keep at a generalization scheme). As mentioned early in this paper, *age* and *occupation* appear to be the attributes where generalization appears to pay off, the former due to its domain and the balance of the mappings among different levels) and the latter due to the structure of its first level that comprises three values only along with a nice balance to the lower-level values, too. The rest of the attributes rise to significant heights before being comparable with *age* and *occupation*.

Table 5.1 Relative Importance of generalization levels for the Adult data set. Left: number of groups and average group size per level; Right: total order of all the levels by relative importance (descending)

	level	num groups	Avg group size	relImp()
age	400000	3455	8.73	1.56
	300000	5380	5.61	1.30
	200000	7015	4.30	1.30
	100000	9117	3.31	1.70
	000000	15537	1.94	0.59
education	040000	8247	3.66	1.26
	030000	10407	2.90	1.30
	020000	13526	2.23	1.09
	010000	14796	2.04	1.05
	000000	15537	1.94	0.95
marital_status	003000	11190	2.70	1.16
	002000	13018	2.32	1.14
	001000	14855	2.03	1.05
	000000	15537	1.94	0.96
occupation	000200	7932	3.80	1.38
	000100	10975	2.75	1.42
	000000	15537	1.94	0.71
race	000020	13478	2.24	1.13
	000010	15210	1.98	1.02
	000000	15537	1.94	0.98
work_class	000003	11790	2.56	1.00
	000002	11798	2.56	1.24
	000001	14668	2.06	1.06
	000000	15537	1.94	0.94

Level	relImp()
age1	1.70
age4	1.56
occupation1	1.42
occupation2	1.38
age3	1.30
education3	1.30
age2	1.30
education4	1.26
work_class2	1.24
marital_status3	1.16
marital_status2	1.14
race2	1.13
education2	1.09
work_class1	1.06
education1	1.05
marital_status1	1.05
race1	1.02
work_class3	1.00
race0	0.98
marital_status0	0.96
education0	0.95
work_class0	0.94
occupation0	0.71
age0	0.59

5.3. The grouping power of lattice nodes and its effect to suppression

Having defined the grouping power and the relative importance of a generalization level, we can now proceed to define the estimated importance of a node. Assuming a node *v*, defined by its quasi-identifier levels as $v[A_1^{h1},A_2^{h2},\ldots A_n^{hn}]$, we exploit the individual metrics of each level and combine them in order to predict the importance of a node in the lattice with respect to its ability to provide as low suppression as possible, if chosen as the elected generalization scheme.

Early failures and level metrics that we have used. In our deliberations, we first started by using the average group size metric as the measure for each level. We tried to provide an estimation of the group size of each node by multiplying the shrinking power of each of its levels and assessed how well the produced metric approximated the actual order of nodes per height with respect to their suppression. This approach did not work very well as the levels with very large group sizes dominated the outcome; so, we decided to change the way we combined the individual metrics and opted for simple summation. Here, observe that the combined metric tries to estimate the goodness of each node as compared to the other nodes in a very efficient way; so, the actual meaning of the produced score is not so important.

The problem with the domination of the scores by large groups came again when we tried to combine the average group size and the relative importance of each node in its hierarchy as the product of these two metrics. So, we decided to use the logarithm of the group size as a useful indication. Overall, the level metrics we have used are:

- Average group size (γ)
- Relative importance of a level (μ)
- The product $\gamma * \mu$
- The product $\log_2(\gamma) * \mu$

Estimated node importance. Given these individual metrics, we define the estimated importance of a node (with respect to its power to reduce suppression) as the sum of the metrics of each of the levels that define the node.

- $\Gamma(v) = \Sigma(\gamma_i)$
- $M(v) = \Sigma\,(\mu_i)$
- $\Gamma M\,(v) = \Sigma\,(\gamma_i * \mu_i)$
- $\Lambda(v) = \Sigma(\log(\gamma_i) * \mu_i)$

For example, assume the case of QI=3 for the Adult data set, with *Age*, *Race*, *WorkClass* as the quasi-identifier set, and the node *A*3*R*2*W*2 (*Age*^{3}, *Race*^{2}, *WorkClass*^{2}) of the resulting lattice. Then, according to the values of Table 5.1, the estimated importance of the node *A*3*R*2*W*2 with the μ metric is 1.30+1.13+1.24=3.67.

Experimental method and findings. We measured the estimated importance of all nodes of the lattice for QI=3,4,5,6 for the Adult data set. For each of the estimator measures, we have measured the 5% top nodes in terms of actual suppression for the case of k-anonymity with k = 3, 10, 50. For heights with too few nodes, we kept at least 2 nodes. The goal of the experiment is to identify which of our prediction metrics provides the best possible approximation to the exact results that the full lattice would give.

For every height *h*, we compare the case of the full lattice and the case of our node selection according to each of the four metrics. According to the nodes selected, we pick the one with the least actual suppression, which we call the winner node for the metric under inspection. Then, we compare this best node per metric with the best actual node and we count the misses we get as well as the deviation of the winner's node suppression against the best possible suppression (of the full lattice).

k=3	Till 30 tuples				Overall			
	#dev(Γ)	Err(Γ)	#dev(Λ)	Err(Λ)	#dev(Γ)	Err(Γ)	#dev(Λ)	Err(Λ)
QI=3	0	0,00%	0	0,00%	3	28,21%	4	26,67%
QI=4	2	21,96%	1	3,41%	5	29,64%	5	11,85%
QI=5	2	2,86%	2	2,35%	4	1,43%	6	7,42%
QI=6	1	0,60%	1	1,31%	4	8,21%	6	8,59%

k=10	Till 30 tuples				Overall			
	#dev(Γ)	Err(Γ)	#dev(Λ)	Err(Λ)	#dev(Γ)	Err(Γ)	#dev(Λ)	Err(Λ)
QI=3	2	52,99%	0	0%	3	22,27%	2	0,36%
QI=4	3	57,87%	1	8,53%	4	29,83%	5	54,56%
QI=5	3	6,84%	2	4,51%	5	3,85%	6	2,54%
QI=6	0	0,00%	2	1,85%	2	5,27%	7	6,34%

k=50	Till 30 tuples				Overall			
	#dev(Γ)	Err(Γ)	#dev(Λ)	Err(Λ)	#dev(Γ)	Err(Γ)	#dev(Λ)	Err(Λ)
QI=3	2	33,20%	2	12,30%	2	16,60%	4	44,01%
QI=4	4	38,38%	1	0,61%	5	22,39%	3	0,36%
QI=5	1	0,05%	1	1,30%	2	0,04%	3	0,90%
QI=6	0	0,00%	2	1,00%	3	24,44%	7	25,08%

Figure 5.1 Number of deviations and accuracy for the estimator functions Γand Λ

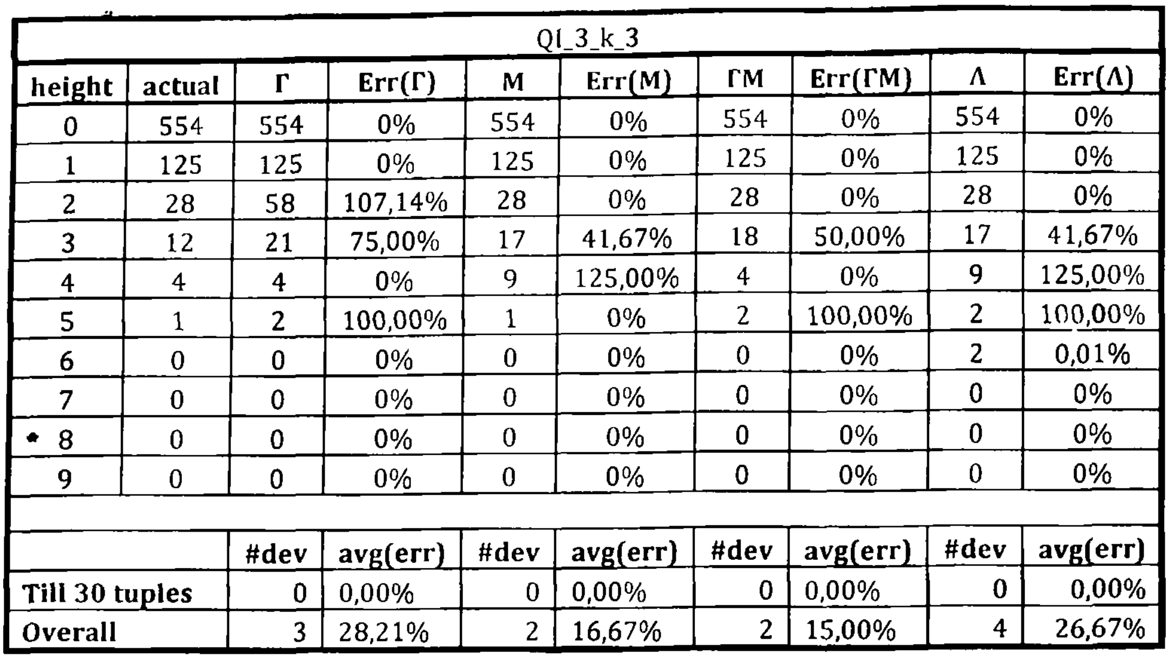
To forestall any possible criticism on the evaluation of selected nodes with respect to their *actual* suppression, we would like to remind the reader that the underlying idea here is as follows: had we used the estimation metric in practice, we would have picked these particular nodes via the metric under inspection and we would have calculated their histogram. Then, a simple exhaustive algorithm could go through all these histograms (that are too few to present a problem) and find the winner per height.

Since the error is the percentage of the difference between the best solutions between full and partial lattice construction over the full lattice's value, we had to handle the cases where the actual suppression was zero. In this case, each extra tuple was given a penalty $|D|^{-1}$, with $|D|$ being the size of the data set. Again, this does not cover adequately all cases as there exist cases where the actual suppression was very small (less than 0.1% of the overall data set) and small differences in the amount of suppression resulted in large errors. So, when we give consolidate results we present one report for the overall experiment and another for the subset of the lattice's heights where the best solution is larger than 0.1% of the data set (in the case of the Adult data set, 30 tuples).

The results are depicted in consolidated form in Figure 5.1 and in detailed form in Figures 5.2-5.13. The consolidated results per combination of *k* and QI size report the number of times that the estimator missed the best possible node (#dev) and the average of the error made by the estimator.

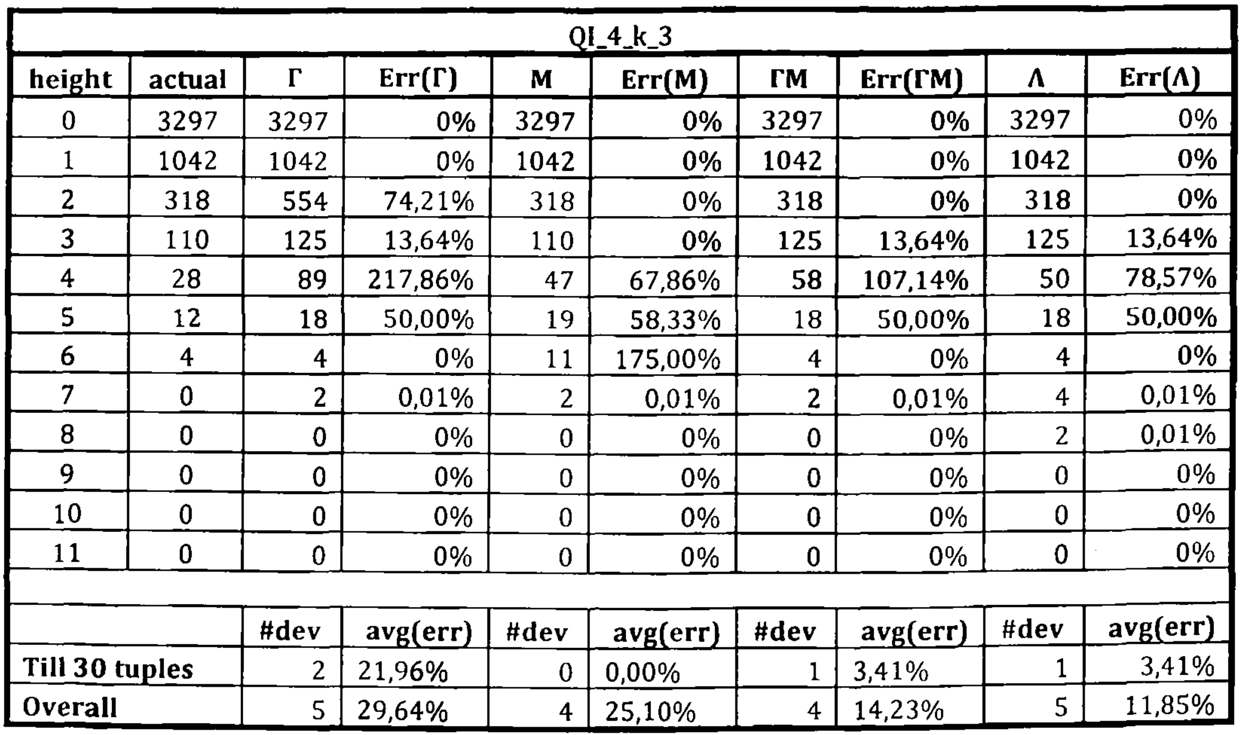
In Figure 5.1 we present the two best estimator methods, the average group size (Γ) and product of the group size's logarithm with its relative importance (Λ). The former, Γ, presents very good results for the largest QI size (6) and a large range of results for the other QI sizes. The latter, Λ, retains a very good estimation range for all occurrences. It is true, however, that its performance drops at the higher level of the lattice, where the best possible suppression is 0; in these cases, Λ frequently misses this possibility, although the selected nodes approximate the best possible solution with very low numbers of suppressed tuples. For completeness, it should be also noted that the relative importance of levels (M) produces frequent misses of the best possible solution, sometimes with significant deviations, whereas the behavior of ΓM follows the one of Γ quite closely – sometimes, with even better results.

Overall, based on all the above, we find Λ *to be the estimator of choice for the subsequent experiments.*



QI_3_k_3									
height	actual	Γ	Err(Γ)	M	Err(M)	ΓM	Err(ΓM)	Λ	Err(Λ)
0	554	554	0%	554	0%	554	0%	554	0%
1	125	125	0%	125	0%	125	0%	125	0%
2	28	58	107,14%	28	0%	28	0%	28	0%
3	12	21	75,00%	17	41,67%	18	50,00%	17	41,67%
4	4	4	0%	9	125,00%	4	0%	9	125,00%
5	1	2	100,00%	1	0%	2	100,00%	2	100,00%
6	0	0	0%	0	0%	0	0%	2	0,01%
7	0	0	0%	0	0%	0	0%	0	0%
8	0	0	0%	0	0%	0	0%	0	0%
9	0	0	0%	0	0%	0	0%	0	0%
		#dev	avg(err)	#dev	avg(err)	#dev	avg(err)	#dev	avg(err)
Till 30 tuples		0	0,00%	0	0,00%	0	0,00%	0	0,00%
Overall		3	28,21%	2	16,67%	2	15,00%	4	26,67%

Figure 5.2 Detailed table of deviation for |QI|=3 and k=3



QI_4_k_3									
height	actual	Γ	Err(Γ)	M	Err(M)	ΓM	Err(ΓM)	Λ	Err(Λ)
0	3297	3297	0%	3297	0%	3297	0%	3297	0%
1	1042	1042	0%	1042	0%	1042	0%	1042	0%
2	318	554	74,21%	318	0%	318	0%	318	0%
3	110	125	13,64%	110	0%	125	13,64%	125	13,64%
4	28	89	217,86%	47	67,86%	58	107,14%	50	78,57%
5	12	18	50,00%	19	58,33%	18	50,00%	18	50,00%
6	4	4	0%	11	175,00%	4	0%	4	0%
7	0	2	0,01%	2	0,01%	2	0,01%	4	0,01%
8	0	0	0%	0	0%	0	0%	2	0,01%
9	0	0	0%	0	0%	0	0%	0	0%
10	0	0	0%	0	0%	0	0%	0	0%
11	0	0	0%	0	0%	0	0%	0	0%
		#dev	avg(err)	#dev	avg(err)	#dev	avg(err)	#dev	avg(err)
Till 30 tuples		2	21,96%	0	0,00%	1	3,41%	1	3,41%
Overall		5	29,64%	4	25,10%	4	14,23%	5	11,85%

Figure 5.3 Detailed table of deviation for |QI|=4 and k=3

QI_5_k_3									
height	**actual**	**Γ**	**Err(Γ)**	**M**	**Err(M)**	**ΓM**	**Err(ΓM)**	**Λ**	**Err(Λ)**
0	10458	10458	0%	10458	0%	10458	0%	10458	0%
1	4514	4514	0%	4514	0%	4514	0%	4514	0%
2	2169	2169	0%	2169	0%	2169	0%	2169	0%
3	1123	1123	0%	1619	44,17%	1123	0%	1123	0,00%
4	716	716	0%	753	5,17%	731	2,09%	731	2,09%
5	322	342	6,21%	377	17,08%	342	6,21%	322	0%
6	108	126	16,67%	164	51,85%	126	16,67%	126	16,67%
7	41	41	0%	47	14,63%	41	0%	41	0%
8	8	8	0%	31	287,50%	8	0%	8	0%
9	2	2	0%	16	700,00%	2	0%	4	100,00%
10	0	2	0,01%	9	0,03%	2	0,01%	2	0,01%
11	0	2	0,01%	0	0%	2	0,01%	2	0,01%
12	0	0	0%	2	0,01%	0	0%	2	0,01%
13	0	0	0%	0	0%	0	0%	0	0%
14	0	0	0%	0	0%	0	0%	0	0%
15	0	0	0%	0	0%	0	0%	0	0%
		#dev	**avg(err)**	**#dev**	**avg(err)**	**#dev**	**avg(err)**	**#dev**	**avg(err)**
Till 30 tuples		2	2,86%	5	16,61%	3	3,12%	2	2,35%
Overall		4	1,43%	9	70,03%	5	1,56%	6	7,42%

Figure 5.4 Detailed table of deviation for |QI|=5 and k=3

QI_6_k_3									
height	actual	Γ	Err(Γ)	M	Err(M)	ΓM	Err(ΓM)	Λ	Err(Λ)
0	15318	15318	0%	15318	0%	15318	0%	15318	0%
1	8304	8304	0%	8304	0%	8304	0%	8304	0%
2	4901	4901	0%	4901	0%	4901	0%	4901	0%
3	2867	2867	0%	4023	40,32%	2867	0%	2867	0%
4	1941	1941	0%	2446	26,02%	2196	13,14%	2196	13,14%
5	1177	1248	6,03%	1177	0%	1177	0%	1177	0%
6	629	629	0%	752	19,55%	629	0%	629	0%
7	354	354	0%	524	48,02%	354	0%	354	0%
8	155	155	0%	243	56,77%	155	0%	155	0%
9	33	33	0%	78	136,36%	33	0%	33	0%
10	9	9	0%	33	266,67%	9	0%	9	0%
11	2	5	150,00%	17	750,00%	5	150,00%	5	150,00%
12	0	1	0,003%	10	0,03%	2	0,01%	4	0,01%
13	0	1	0,003%	2	0,01%	2	0,01%	2	0,01%
14	0	0	0%	0	0%	0	0%	2	0,01%
15	0	0	0%	0	0%	2	0,01%	2	0,01%
16	0	0	0%	0	0%	0	0%	0	0%
17	0	0	0%	0	0%	0	0%	0	0%
18	0	0	0%	0	0%	0	0%	0	0%
		#dev	avg(err)	#dev	avg(err)	#dev	avg(err)	#dev	avg(err)
Till 30 tuples		1	0,60%	6	32,71%	1	1,31%	1	1,31%
Overall		4	8,21%	10	70,72%	5	8,59%	6	8,59%

Figure 5.5 Detailed table of deviation for |QI|=6 and k=3

QI 3 k 10									
height	actual	Γ	Err(Γ)	M	Err(M)	ΓM	Err(ΓM)	Λ	Err(Λ)
0	1921	1921	0%	1921	0%	1921	0%	1921	0%
1	522	522	0%	522	0%	522	0%	522	0%
2	170	257	51,18%	170	0%	170	0%	170	0%
3	51	133	160,78%	51	0%	51	0%	51	0%
4	28	31	10,71%	29	3,57%	31	10,71%	29	3,57%
5	2	2	0%	14	600,00%	2	0%	2	0%
6	0	0	0%	0	0%	0	0%	2	0,01%
7	0	0	0%	0	0%	0	0%	0	0%
8	0	0	0%	0	0%	0	0%	0	0%
9	0	0	0%	0	0%	0	0%	0	0%
		#dev	avg(err)	#dev	avg(err)	#dev	avg(err)	#dev	avg(err)
Till 30 tuples		2	52,99%	0	0%	0	0%	0	0%
Overall		3	22,27%	2	60,36%	1	1,07%	2	0,36%

Figure 5.6 Detailed table of deviation for |QI|=3 and k=10

[illegible]									
height	actual	Γ	Err(Γ)	M	Err(M)	ΓM	Err(ΓM)	Λ	Err(Λ)
0	9416	9416	0%	9416	0%	9416	0%	9416	0%
1	3273	3273	0%	3273	0%	3273	0%	3273	0%
2	1261	1921	52,34%	1261	0%	1261	0%	1261	0%
3	522	522	0%	752	44,06%	522	0%	522	0%
4	170	378	122,35%	285	67,65%	257	51,18%	257	51,18%
5	51	139	172,55%	61	19,61%	139	172,55%	51	0%
6	28	31	10,71%	29	3,57%	31	10,71%	29	3,57%
7	2	2	0%	14	600,00%	2	0%	14	600,00%
8	0	0	0%	0	0%	0	0%	2	0,01%
9	0	0	0%	2	0,01%	0	0%	2	0,01%
10	0	0	0%	0	0%	0	0%	0	0%
11	0	0	0%	0	0%	0	0%	0	0%
		#dev	avg(err)	#dev	avg(err)	#dev	avg(err)	#dev	avg(err)
Till 30 tuples		3	57,87%	3	21,89%	2	37,29%	1	8,53%
Overall		4	29,83%	6	61,24%	3	19,54%	5	54,56%

Figure 5.7 Detailed table of deviation for |QI|=4 and k=10

QI=5,k=10

height	actual	Γ	Err(Γ)	M	Err(M)	ΓM	Err(ΓM)	Λ	Err(Λ)
0	18916	18916	0%	18916	0%	18916	0%	18916	0%
1	10944	10944	0%	10944	0%	10944	0%	10944	0%
2	6151	6151	0%	6151	0%	6151	0%	6151	0%
3	3468	3468	0%	4824	39,10%	3468	0%	3468	0%
4	2065	2065	0%	2868	38,89%	2508	21,45%	2508	21,45%
5	1160	1207	4,05%	2007	73,02%	1207	4,05%	1160	0%
6	578	578	0%	1004	73,70%	578	0%	578	0%
7	230	274	19,13%	230	0%	274	19,13%	274	19,13%
8	60	83	38,33%	141	135,00%	83	38,33%	60	0%
9	14	14	0%	41	192,86%	14	0%	14	0%
10	0	14	0,05%	22	0,07%	14	0,05%	14	0,05%
11	0	2	0,01%	8	0,03%	2	0,01%	2	0,01%
12	0	0	0%	14	0,05%	0	0%	2	0,01%
13	0	0	0%	4	0,01%	2	0,01%	2	0,01%
14	0	0	0%	0	0%	0	0%	0	0%
15	0	0	0%	0	0%	0	0%	0	0%

	#dev	avg(err)	#dev	avg(err)	#dev	avg(err)	#dev	avg(err)
Till 30 tuples	3	6,84%	5	39,97%	4	9,22%	2	4,51%
Overall	5	3,85%	10	34,55%	7	5,19%	6	2,54%

Figure 5.8 Detailed table of deviation for |QI|=5 and k=10

QI_6_k_10									
height	actual	Γ	Err(Γ)	M	Err(M)	ΓM	Err(ΓM)	Λ	Err(Λ)
0	24040	24040	0%	24040	0%	24040	0%	24040	0%
1	15836	15836	0%	15836	0%	15836	0%	15836	0%
2	10649	10649	0%	10649	0%	10649	0%	10649	0%
3	7153	7153	0%	9200	28,62%	7153	0%	7153	0%
4	5063	5063	0%	6236	23,17%	5827	15,09%	5827	15,09%
5	3562	3562	0%	3616	1,52%	3562	0%	3562	0%
6	1823	1823	0%	2660	45,91%	1823	0%	1823	0%
7	1222	1222	0%	1895	55,07%	1222	0%	1222	0%
8	639	639	0%	970	51,80%	639	0%	639	0%
9	285	285	0%	493	72,98%	285	0%	300	5,26%
10	54	54	0%	188	248,15%	54	0%	54	0%
11	21	21	0%	73	247,62%	21	0%	21	0%
12	7	14	100,00%	29	314,29%	14	100,00%	14	100,00%
13	0	14	0,05%	14	0,05%	14	0,05%	14	0,05%
14	0	0	0%	8	0,03%	2	0,01%	14	0,05%
15	0	0	0%	14	0,05%	2	0,01%	2	0,01%
16	0	0	0%	4	0,01%	2	0,01%	2	0,01%
17	0	0	0%	0	0%	0	0%	0	0%
18	0	0	0%	0	0%	0	0%	0	0%
		#dev	avg(err)	#dev	avg(err)	#dev	avg(err)	#dev	avg(err)
Till 30 tuples		0	0,00%	8	47,93%	1	1,37%	2	1,85%
Overall		2	5,27%	14	57,33%	6	6,06%	7	6,34%

Figure 5.9 Detailed table of deviation for |QI|=6 and k=10

QI_3_k_50									
height	actual	Γ	Err(Γ)	M	Err(M)	ΓM	Err(ΓM)	Λ	Err(Λ)
0	8297	8297	0%	8297	0%	8297	0%	8297	0%
1	2123	2123	0%	2123	0%	2123	0%	2123	0%
2	1345	1345	0%	1402	4,24%	1402	4,24%	1402	4,24%
3	428	698	63,08%	673	57,24%	698	63,08%	673	57,24%
4	137	278	102,92%	137	0%	278	102,92%	137	0%
5	14	14	0%	76	442,86%	14	0%	67	378,57%
6	14	14	0%	14	0%	14	0%	14	0%
7	0	0	0%	14	0,05%	14	0,05%	14	0,05%
8	0	0	0%	0	0%	0	0%	0	0%
9	0	0	0%	0	0%	0	0%	0	0%
		#dev	avg(err)	#dev	avg(err)	#dev	avg(err)	#dev	avg(err)
Till 30 tuples		2	33,20%	2	12,30%	3	34,05%	2	12,30%
Overall		2	16,60%	4	50,44%	4	17,03%	4	44,01%

Figure 5.10 Detailed table of deviation for |QI|=3 and k=50

QI_4_k_50									
height	actual	Γ	Err(Γ)	M	Err(M)	ΓM	Err(ΓM)	Λ	Err(Λ)
0	20066	20066	0%	20066	0%	20066	0,00%	20066	0%
1	10053	10053	0%	10053	0%	10053	0,00%	10053	0%
2	4612	7036	52,56%	4612	0%	4612	0,00%	4612	0%
3	2123	2123	0%	3217	51,53%	2123	0,00%	2123	0%
4	1345	1848	37,40%	1555	15,61%	1402	4,24%	1402	4,24%
5	359	631	75,77%	359	0,00%	631	75,77%	359	0%
6	137	278	102,92%	137	0,00%	278	102,92%	137	0%
7	14	14	0%	76	442,86%	14	0,00%	14	0%
8	0	14	0%	14	0,05%	14	0,05%	14	0,05%
9	0	0	0%	14	0,05%	14	0,05%	14	0,05%
10	0	0	0%	0	0%	0	0%	0	0%
11	0	0	0%	0	0%	0	0%	0	0%
		#dev	avg(err)	#dev	avg(err)	#dev	avg(err)	#dev	avg(err)
Till 30 tuples		4	38,38%	2	9,59%	3	26,13%	1	0,61%
Overall		5	22,39%	5	42,51%	5	15,25%	3	0,36%

Figure 5.11 Detailed table of deviation for |QI|=4 and k=50

QI_5_k_50									
height	actual	Γ	Err(Γ)	M	Err(M)	ΓM	Err(ΓM)	Λ	Err(Λ)
0	29650	29650	0%	29650	0%	29650	0%	29650	0%
1	19750	19750	0%	19750	0%	19750	0%	19750	0%
2	14575	14575	0%	14575	0%	14575	0%	14575	0%
3	10546	10546	0%	12933	22,63%	10546	0%	10546	0%
4	6954	6954	0%	8073	16,09%	7328	5,38%	7947	14,28%
5	4336	4336	0%	6870	58,44%	4336	0%	4336	0%
6	2002	2002	0%	5014	150,45%	2002	0%	2002	0%
7	1366	1366	0%	2243	64,20%	1366	0%	1366	0%
8	613	613	0%	1068	74,23%	613	0%	613	0%
9	169	170	0,59%	234	38,46%	170	0,59%	169	0%
10	59	59	0%	137	132,20%	59	0%	59	0%
11	14	14	0%	14	0%	14	0%	14	0%
12	0	14	0,05%	14	0,05%	14	0,05%	14	0,05%
13	0	0	0%	14	0,05%	14	0,05%	14	0,05%
14	0	0	0%	0	0%	0	0%	0	0%
15	0	0	0%	0	0%	0	0%	0	0%
		#dev	avg(err)	#dev	avg(err)	#dev	avg(err)	#dev	avg(err)
Till 30 tuples		1	0,05%	8	50,61%	2	0,54%	1	1,30%
Overall		2	0,04%	10	34,80%	4	0,38%	3	0,90%

Figure 5.12 Detailed table of deviation for |QI|=5 and k=50

QI_6_k_50									
height	**actual**	**Γ**	**Err(Γ)**	**M**	**Err(M)**	**ΓM**	**Err(ΓM)**	**Λ**	**Err(Λ)**
0	29868	29868	0%	29868	0%	29868	0%	29868	0%
1	24626	24626	0%	24626	0%	24626	0%	24626	0%
2	19084	19084	0%	19084	0%	19084	0%	19084	0%
3	15380	15380	0%	18364	19,40%	15380	0%	15380	0%
4	12278	12278	0%	13893	13,15%	13156	7,15%	13630	11,01%
5	9088	9088	0%	9855	8,44%	9088	0%	9176	0,97%
6	5515	5515	0%	8444	53,11%	5515	0%	5515	0%
7	4360	4360	0%	6684	53,30%	4360	0%	4360	0%
8	2482	2482	0%	3959	59,51%	2482	0%	2482	0%
9	1786	1786	0%	2449	37,12%	1786	0%	1786	0%
10	869	869	0%	1169	34,52%	869	0%	869	0%
11	137	137	0%	515	275,91%	137	0%	137	0%
12	14	79	464,29%	234	1571,43%	79	464,29%	79	464,29%
13	0	14	0,05%	14	0,05%	14	0,05%	14	0,05%
14	0	0	0%	14	0,05%	14	0,05%	14	0,05%
15	0	14	0,05%	14	0,05%	14	0,05%	14	0,05%
16	0	0	0%	14	0,05%	14	0,05%	14	0,05%
17	0	0	0%	0	0%	0	0%	0	0%
18	0	0	0%	0	0%	0	0%	0	0%
		#dev	**avg(err)**	**#dev**	**avg(err)**	**#dev**	**avg(err)**	**#dev**	**avg(err)**
Till 30 tuples		0	0,00%	9	46,21%	1	0,60%	2	1,00%
Overall		3	24,44%	14	111,90%	6	24,82%	7	25,08%

Figure 5.13 Detailed table of deviation for |QI|=6 and k=50

5.4. Preprocessing time

The time to complete the preprocessing time for the different QI sizes of the Adult data set is depicted in Fig. 5.14. We observe that the time falls to approximately one minute for QI = 6 (remember that it used to be approximately 20 minutes for the full lattice, which demonstrates a linear speedup with the approximation factor.

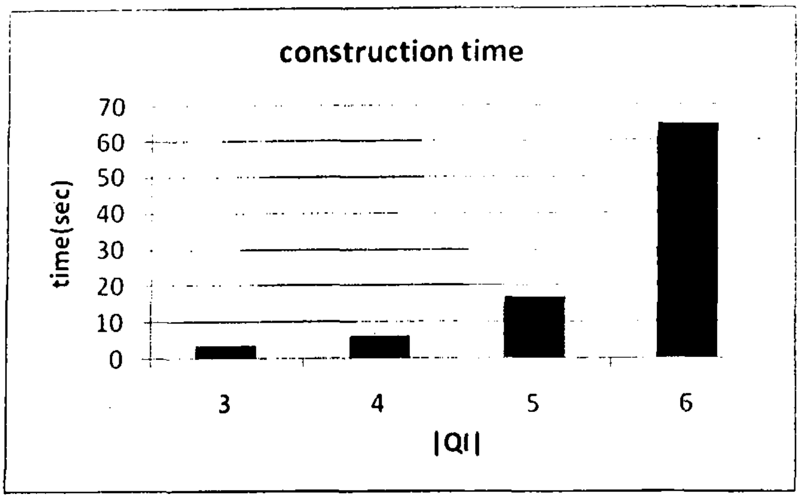


Figure 5.14 Total constuction time for the partial lattice of the Adult data set

We can observe the exponential curve in the construction time as QI increases. This is clearly due to the nature of the problem: as we keep the percentage of nodes fixed (here: 5%) every extra attribute in the QI scales up the number of nodes by the size of its levels.

The breakdown of the lattice and histogram construcion time is as listed in Table 5.2 for the Adult data set and Table 5.3 for the IPUMS data set.

Table 5.2 Breakdown of construction time (sec) of partial lattice for the Adult data set

	QI=3	QI=4	QI=5	QI=6
Level importance comp.	1.15	1.8	3.27	5.24
Node importance comp.	0.05	0.18	0.40	0.99
Histogram computation	2.52	4.25	13.32	58.47
Overall	3.72	6.24	17.00	64.70

We observe that, as expected, the interaction with the database is the one that consumes most of the time.The first of these interactions, specifically the computation of level importance requires one aggregate query per level and does not take too much time. At the same time, the computation of histograms is largely affected by the number of nodes selected to be part of the partial lattice; since our rule indicates a fixed percentage, the exponential nature of the cost remains, albeit significantly reduced.

Table 5.3 Breakdown of construction time (msec) of partial lattice for the IPUMS data set

	QI=4
Level importance comp.	31,64
Node importance comp.	0,18
Histogram computation	77,65
Overall	109,47

5.4.1. Answering user requests via the partial lattice

The method for answering a request by the user is a modified variant of the algorithm of the full lattice.

```
Algorithm PartialLatticeAnonymityNegotiation(L,k,h,MaxSupp)
In: Partial lattice L with the histograms for R,H, constraints for k, h, MaxSupp
Out: an exact solution s[v,k,h,supp] or s1,s2,s3, si=[v_i,k_i,h_i,supp_i]
Begin
Let v_max be the node that corresponds to the constraint h ;
if v_max is part of L{
        Check v_max for an exact answer;
        if no such answer exists, goto Appr;
}
for every height h, in height(v_max)-1, down to 0{
        for every node v in h, v in descendants(v_max), {
            if an exact answer is given by v
                keep the v with the minimum suppression as v_opt;
                (break ties by h)
        }//observe: all descendants of v_max must be checked in all levels
}
if an exact answer is found,
        return v_opt with its answer;
else{
Appr:   for every height h, in height(v_max) down to 0{
            for every node v in desc(v_max) in h {
                check suppressed(v,k);
                keep v_opt$^M$ the node with the least suppression, k respected;
                check max k for v, s.t., MaxSupp is respected;
                keep v_opt$^k$ the node with the max k that respects MaxSupp;
            }
        }
        approxSol_1=solution(v_opt$^k$)
        approxSol_3=solution(v_opt$^M$)
        approxSol_2=ApproximateH(L,v_max,height(v_max),height(top),k,h,MaxSupp);
        return approxSol_1, approxSol_2, approxSol_3;
}
End.
```

Figure 5.15 Algorithm for Partial lattice Anonymity Negotiation

The algorithm starts with a quick check: if the highest node of the sublattice of valid answers, v_max, cannot return an exact answer, then it is clear that no other node in the sub-lattice can; therefore the search is directed towards finding a set of approximate answers (Line 15-28). If however, the node v_max is not part of the partial lattice or, it is part of the partial lattice and gives an exact answer, then, the sublattice must be checked. Due to the small size of the sublattice, it is quite reasonable to explore it via a practically exhaustive search (Lines 6–12). So, we search all levels and keep the answer with the minimum suppression (ties over suppression are broken by picking the solution with the least height). Here, due to the fact that the lattice is partial, we must note that we cannot rely on any pruning criteria: if a node fails to give an exact answer at height *h*, this does not mean that there are no nodes in heights lower than *h* that can answer. Therefore, the entire sublattice must be searched. Observe also, that due to the constraint that at least two nodes per level are computed, the bottom node is always computed; so, there is always at least one descendant of v_max in the partial lattice *L*.

If an exact answer is not found, we must search for approximate answers. The two of the three approximations are performed in a simple way: we search all nodes in the heights from v_max to 0 to find (a) the node that gives the least suppression, keeping k fixed, and (b) the node that gives the maximum k, keeping MaxSupp fixed. This is shown in lines 16 – 23. Apart from these two suggestions, we need to find the node with the least height that respects both k and MaxSupp. This is done the same way as in the full lattice (Line 26) --see also function *ApproximateH*, which we summarize here: we search the upper part of the lattice with binary search; if a node answers positively we search downwards for lower nodes that can answer too; else we search upwards and check if the level under investigation is unable to provide a solution.

5.5. Quality of solution

Having explained the method via which user requests are answered, we can now proceed to discuss our findings concerning the quality of answers returned by the algorithm of the previous subsection. Figures 5.16 – 5.18 depict the detailed results of the workloads of section 4 when the partial lattice is used instead of the full one.

Parameters	Actual solution			Aproximate solution			QoS		
	k	supp	height	k	supp	height	d(k)	d(supp)	d(h)
					Q1				
3, 321, 2 1 1	3	125	1	3	125	1	0	0	0
10, 321, 2 1 1	10	170	2	10	170	2	0	0	0
50, 321, 2 1 1				No exact answer. Approx:					
				50	673	3			
	50	251	4	50	137	4	0	114	0
				31	314	3			
					[illegible]				
3, 321, 2 0 1 1				No exact answer. Approx:					
				3	635	2			
	3	283	3	3	50	4	0	233	1
10, 321, 2 0 1 1	10	655	4	10	2349	2	0	1694	-2
	10	170	4	10	257	4	0	87	0
	5	301	4	No solution					
50, 321, 2 0 1 1	50	4077	4	50	7669	2	0	3592	-2
	50	137	6	50	137	6	0	0	0
	5	301	4	No solution					
					[illegible]				
3, 321, 2 1 0 1 2	3	656	6	3	3639	2	0	2983	-4
	3	108	6	3	126	6	0	18	0
	No solution			No solution					
10, 321, 2 1 0 1 2	10	2533	6	10	9223	2	0	6690	-4
	10	230	7	10	274	7	0	44	0
	No solution			No solution					
50, 321, 2 1 0 1 2	50	9214	6	50	19324	2	0	10110	-4
	50	169	9	50	169	9	0	0	0
	No solution			No solution					
					[illegible]				
3, 321, 2 1 1 0 1 2	3	1611	7	3	7226	2	0	5615	-5
	3	155	8	3	155	8	0	0	0
	No solution			No solution					
10, 321, 2 1 1 0 1 2	10	5362	7	10	14627	2	0	9265	-5
	10	285	9	10	300	9	0	15	0
	No solution			No solution					
50, 321, 2 1 1 0 1 2	50	15106	7	50	24558	2	0	9452	-5
	50	137	11	50	137	11	0	0	0
	No solution			No solution					

Figure 5.16 Qos in details for Variant k

Parameters	*Actual solution*			*Aproximate solution*			QoS		
	k	supp	height	k	supp	height	d(k)	d(supp)	d(h)
	[illegible]								
10, 321, 1 0 1	10	257	2	10	257	2	0	0	0
10, 321, 2 1 1	10	170	2	10	170	2	0	0	0
10, 321, 2 1 2	10	170	2	10	170	2	0	0	0
	[illegible]								
10, 321, 1 0 0 1	10	2349	2	10	2349	2	0	0	0
	10	170	4	10	257	4	0	87	0
	No solution			No solution					
10, 321, 2 0 1 1	10	655	4	55	10	2349	0	1694	-2
	10	170	4	61	10	257	0	87	0
	5	301	4	No solution			5	301	-4
10, 321, 2 1 1 2	10	285	4	10	285	4	0	0	0
	[illegible]								
10, 321, 1 1 0 0 1	10	8169	3	10	9223	2	0	1054	-1
	10	230	7	10	274	7	0	44	0
	No solution			No solution					
10, 321, 2 1 0 1 2	10	2533	6	10	9223	2	0	6690	-4
	10	230	7	10	274	7	0	44	0
	No solution			No solution					
10, 321, 2 2 1 1 2	10	369	8	10	2250	5	0	1881	-3
	10	60	8	10	60	8	0	0	0
	9	315	8	No solution			9	315	-8
	[illegible]								
10, 321, 1 1 1 0 0 1	10	12823	4	10	14627	2	0	1804	-2
	10	285	9	10	300	9	0	15	0
	No solution			No solution					
10, 321, 2 1 1 0 1 2	10	5262	7	10	14627	2	0	9365	-5
	10	285	9	10	300	9	0	15	0
	No solution			No solution					
10, 321, 2 2 2 1 1 2	10	712	10	10	3971	6	0	3259	-4
	10	54	10	10	54	10	0	0	0
	5	298	10	No solution			5	298	-10

Figure 5.17 Qos in details for Variant level

Parameters	*Actual solution*			*Aproximate solution*			QoS		
	k	supp	height	k	supp	height	d(k)	d(supp)	d(h)
	|O|=3								
10, 32, 2 1 1	10	55	4	10	77	3	0	22	-1
	10	28	4	10	29	4	0	1	0
	7	31	4	5	32	3	-2	1	-1
10, 321, 2 1 1	10	170	2	10	170	2	0	0	0
10, 3216, 2 1 1	10	1921	0	10	1921	0	0	0	0
	[illegible]								
10, 32, 2 0 1 1	65	10	655	10	2349	2	0	1694	-2
	35	10	28	10	29	6	0	1	0
	No Solution			No Solution					
10, 321, 2 0 1 1	10	655	4	10	2349	2	0	1694	-2
	10	170	4	10	257	4	0	87	0
	5	301	4	No Solution			5	301	-4
10, 3216, 2 0 1 1	10	2110	2	10	2349	2	0	239	0
	[illegible]								
10, 32, 2 1 0 1 2	10	2533	6	10	9223	2	0	6690	-4
	10	14	9	10	14	9	0	0	0
	No Solution			No Solution					
10, 321, 2 1 0 1 2	10	2533	6	10	9223	2	0	6690	-4
	10	230	7	10	274	7	0	44	0
	No Solution			No Solution					
10, 3216, 2 1 0 1 2				No exact answer. Approx:					
				10	9223	2			
	551	10	2533	10	578	6	0	1955	0
				2	2211	2			
	|O|=6								
10, 32, 2 1 1 0 1 2	10	5362	7	10	14627	2	0	9265	-5
	10	21	11	10	21	11	0	0	0
	No Solution			No Solution					
10, 321, 2 1 1 0 1 2	10	5362	7	10	14627	2	0	9265	-5
	10	285	9	10	300	9	0	15	0
	No Solution			No Solution					
10, 3216, 2 1 1 0 1 2	10	5362	7	10	14627	2	0	9265	-5
	10	1222	7	10	1222	7	0	0	0
	5	2915	7	No Solution			5	2915	-7

Figure 5.18 Qos in details for Variant max supp

The detailed Tables do not reveal too much per se; we complement them with Tables 5.19 – 5.21 where we discuss each answering method (i.e., exact answers along with the 3 relaxations) in isolation. In all these Tables, the cells with grey background are the ones where the full lattice allowed the derivation of an exact solution and the partial lattice failed. The cells with the blue background are the ones where the respective phenomenon occurred for relaxation 3. Remember that:

- Approximation 1 keeps *k* and **h** fixed and tries to find the closest possible suppression that the data set can provide
- Approximation 2 keeps *k* and *maxSupp* fixed and tries to find the lowest possible level where both these values are respected
- Approximation 3 keeps *maxSupp* and **h** fixed and tries to find the closest possible *k* to the original one that the data set can support

The overall performance of the partial lattice with just a 5% support of the full lattice's nodes seems quite satisfactory.

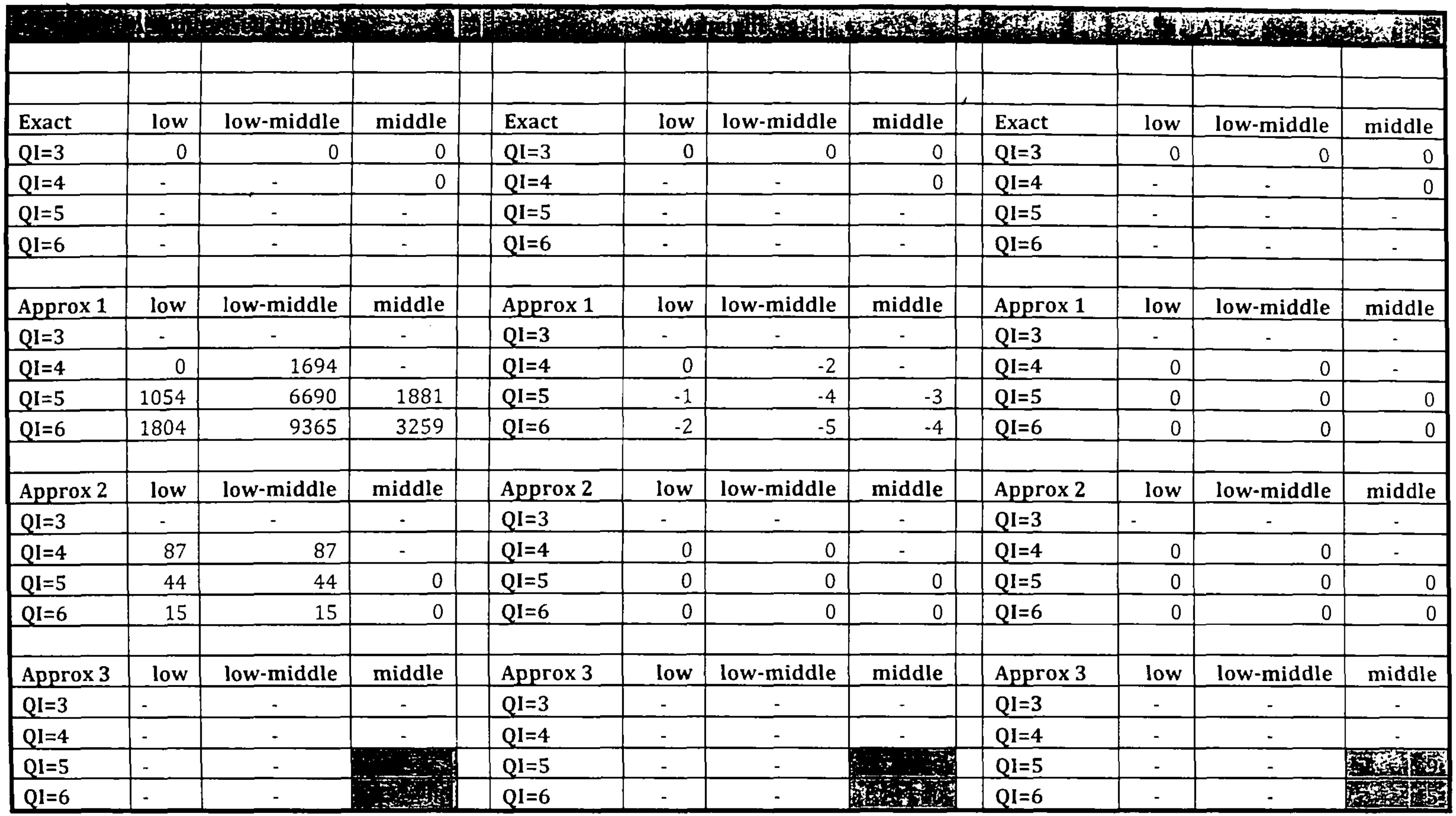
- Approximation 1 has the tendency to move downwards the lattice, until a node that is within the sublattice of vmax is found. So, all the solutions are quite lower than the height constraint (with a difference ranging between -1 and -5) and therefore provide significantly larger suppressions than the one suggested by the full lattice. Remember, however that this is just a suggestion in the context of an interactive user session.
- Approximation 2 tries to minimize the height that provides a solution that respects both k and maxSupp and apparently it does a pretty good job in all occasions (see all three tables for column Δheight and section Approximation 2 in all three tables) with small deviations for the suppressed tuples (but still, within the user's threshold) and no deviations for k with respect to the answer of the full lattice. Remember that this is the most complicated search as it travels throughout the whole lattice in search for an answer.
- Approximation 3 fails frequently in both the full and the partial lattice. Unfortunately, the partial lattice fails to support this approximation. Out of the 36 possible value combinations, 9 had an exact answer in both the full and the partial lattice(so the approximation never fired in the first place) and out of the 27 remaining cases, (a) 17 cases presented no solution in neither the full nor

the partial lattice, (b) 6 cases had an exact answer in the full lattice and no solution in the partial lattice, (c) 3 cases had an answer in the full lattice and an approximation in the partial lattice, and (d) only 1 case had an approximation in the both lattices.

Overall, we practically had three occurrences where the full lattice gives an exact answer and the partial lattice fails: (i) QI=3, k=50, (ii) QI=4, k=3 (iii) QI=5, maxSupp=3126. We find this performance quite satisfactory.

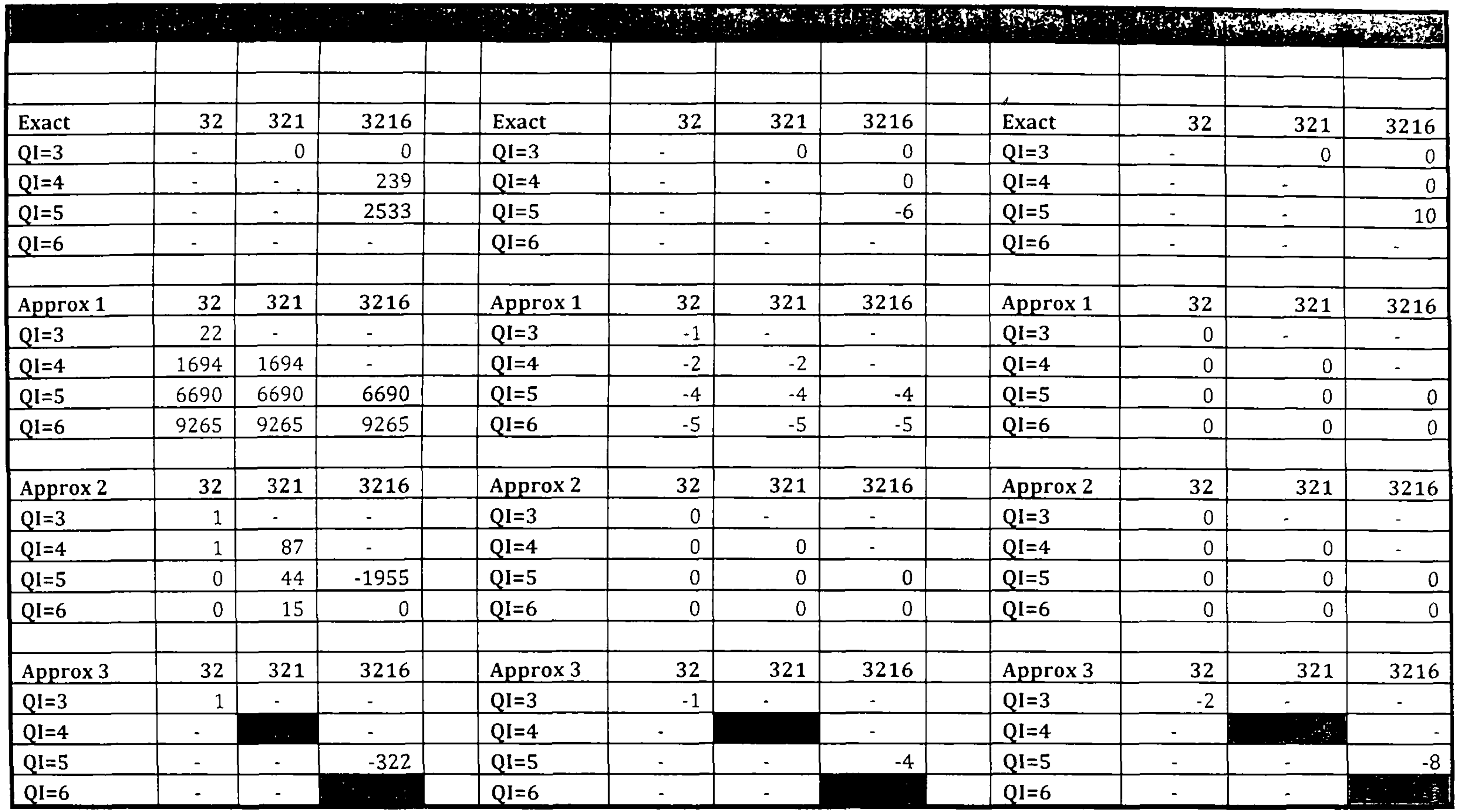
Exact	k=3	k=10	k=50		Exact	k=3	k=10	k=50		Exact	k=3	k=10	k=50
QI=3	0	0	251		QI=3	0	0	4		QI=3	0	0	50
QI=4	283	-	-		QI=4	3	-	-		QI=4	3	-	-
QI=5	-	-	-		QI=5	-	-	-		QI=5	-	-	-
QI=6	-	-	-		QI=6	-	-	-		QI=6	-	-	-
Approx 1	k=3	k=10	k=50		Approx 1	k=3	k=10	k=50		Approx 1	k=3	k=10	k=50
QI=3	-	-	422		QI=3	-	-	-1		QI=3	-	-	0
QI=4	-352	1694	3592		QI=4	-1	-2	-2		QI=4	0	0	0
QI=5	2983	6690	10110		QI=5	-4	-4	-4		QI=5	0	0	0
QI=6	5615	9265	9452		QI=6	-5	-5	-5		QI=6	0	0	0
Approx 2	k=3	k=10	k=50		Approx 2	k=3	k=10	k=50		Approx 2	k=3	k=10	k=50
QI=3	-	-	-114		QI=3	-	-	0		QI=3	-	-	0
QI=4	-233	87	0		QI=4	1	0	0		QI=4	0	0	0
QI=5	18	44	0		QI=5	0	0	0		QI=5	0	0	0
QI=6	0	15	0		QI=6	0	0	0		QI=6	0	0	0
Approx 3	k=3	k=10	k=50		Approx 3	k=3	k=10	k=50		Approx 3	k=3	k=10	k=50
QI=3	-	-	63		QI=3	-	-	-1		QI=3	-	-	-19
QI=4	283				QI=4	3				QI=4	3		
QI=5	-	-	-		QI=5	-	-	-		QI=5	-	-	-
QI=6	-	-	-		QI=6	-	-	-		QI=6	-	-	-

Figure 5.19 Summary of QoS deterioration for variant k



[illegible]				[illegible]				[illegible]			
Exact	low	low-middle	middle	Exact	low	low-middle	middle	Exact	low	low-middle	middle
QI=3	0	0	0	QI=3	0	0	0	QI=3	0	0	0
QI=4	-	-	0	QI=4	-	-	0	QI=4	-	-	0
QI=5	-	-	-	QI=5	-	-	-	QI=5	-	-	-
QI=6	-	-	-	QI=6	-	-	-	QI=6	-	-	-
Approx 1	low	low-middle	middle	Approx 1	low	low-middle	middle	Approx 1	low	low-middle	middle
QI=3	-	-	-	QI=3	-	-	-	QI=3	-	-	-
QI=4	0	1694	-	QI=4	0	-2	-	QI=4	0	0	-
QI=5	1054	6690	1881	QI=5	-1	-4	-3	QI=5	0	0	0
QI=6	1804	9365	3259	QI=6	-2	-5	-4	QI=6	0	0	0
Approx 2	low	low-middle	middle	Approx 2	low	low-middle	middle	Approx 2	low	low-middle	middle
QI=3	-	-	-	QI=3	-	-	-	QI=3	-	-	-
QI=4	87	87	-	QI=4	0	0	-	QI=4	0	0	-
QI=5	44	44	0	QI=5	0	0	0	QI=5	0	0	0
QI=6	15	15	0	QI=6	0	0	0	QI=6	0	0	0
Approx 3	low	low-middle	middle	Approx 3	low	low-middle	middle	Approx 3	low	low-middle	middle
QI=3	-	-	-	QI=3	-	-	-	QI=3	-	-	-
QI=4	-	-	-	QI=4	-	-	-	QI=4	-	-	-
QI=5	-	-	[illegible]	QI=5	-	-	[illegible]	QI=5	-	-	[illegible]
QI=6	-	-	[illegible]	QI=6	-	-	[illegible]	QI=6	-	-	[illegible]

Figure 5.20 Summary of QoS deterioration for variant height



Exact	32	321	3216		Exact	32	321	3216		Exact	32	321	3216
QI=3	-	0	0		QI=3	-	0	0		QI=3	-	0	0
QI=4	-	-	239		QI=4	-	-	0		QI=4	-	-	0
QI=5	-	-	2533		QI=5	-	-	-6		QI=5	-	-	10
QI=6	-	-	-		QI=6	-	-	-		QI=6	-	-	-
Approx 1	32	321	3216		Approx 1	32	321	3216		Approx 1	32	321	3216
QI=3	22	-	-		QI=3	-1	-	-		QI=3	0	-	-
QI=4	1694	1694	-		QI=4	-2	-2	-		QI=4	0	0	-
QI=5	6690	6690	6690		QI=5	-4	-4	-4		QI=5	0	0	0
QI=6	9265	9265	9265		QI=6	-5	-5	-5		QI=6	0	0	0
Approx 2	32	321	3216		Approx 2	32	321	3216		Approx 2	32	321	3216
QI=3	1	-	-		QI=3	0	-	-		QI=3	0	-	-
QI=4	1	87	-		QI=4	0	0	-		QI=4	0	0	-
QI=5	0	44	-1955		QI=5	0	0	0		QI=5	0	0	0
QI=6	0	15	0		QI=6	0	0	0		QI=6	0	0	0
Approx 3	32	321	3216		Approx 3	32	321	3216		Approx 3	32	321	3216
QI=3	1	-	-		QI=3	-1	-	-		QI=3	-2	-	-
QI=4	-		-		QI=4	-		-		QI=4	-		-
QI=5	-	-	-322		QI=5	-	-	-4		QI=5	-	-	-8
QI=6	-	-			QI=6	-	-			QI=6	-	-	

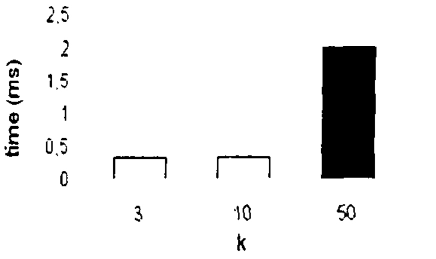
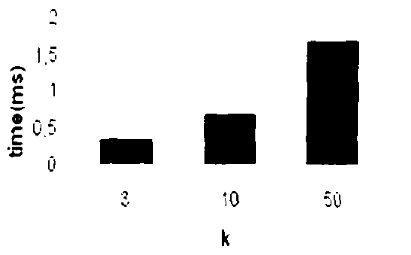
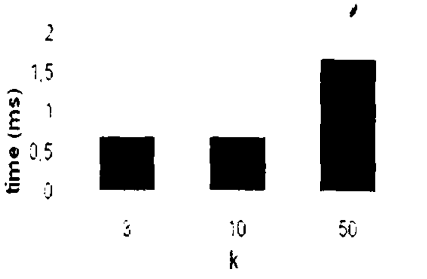
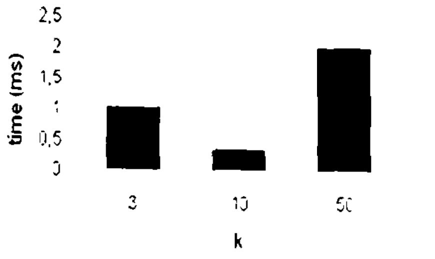
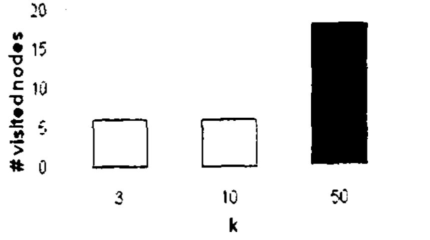
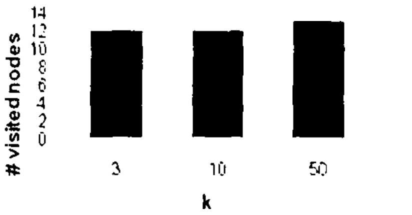
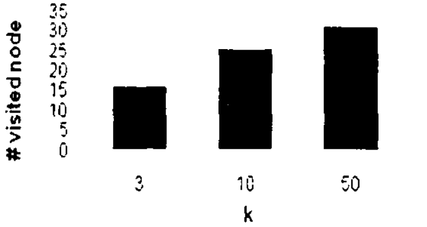
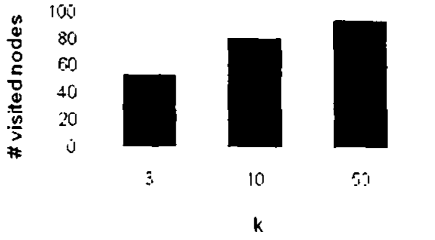
Figure 5.21 Summary of QoS deterioration for variant maxSupp

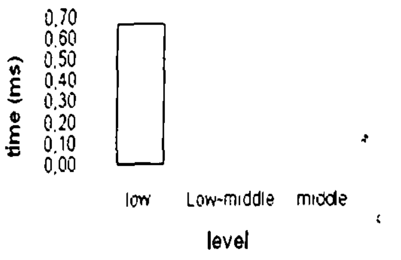
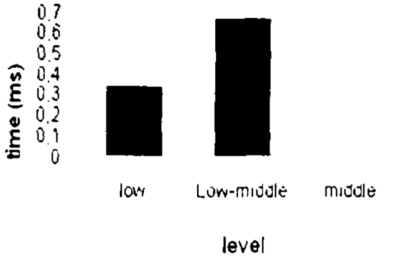
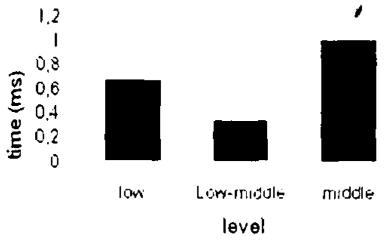
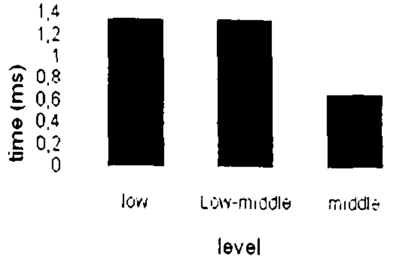
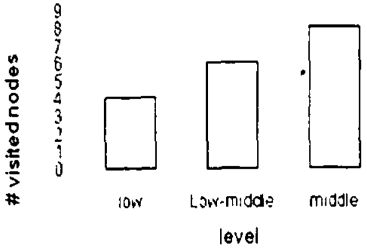
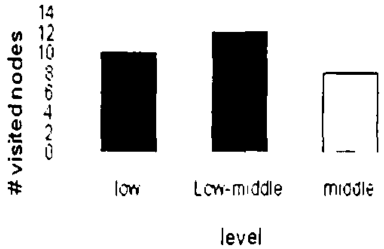
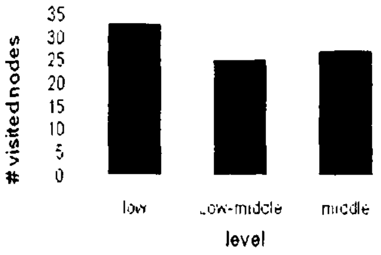
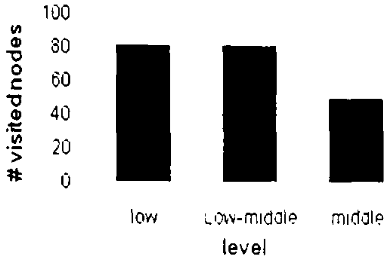
5.6. Performance of Algorithm PartialLatticeNegotiation

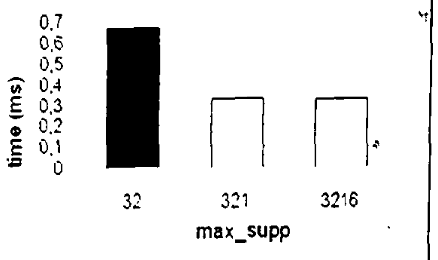
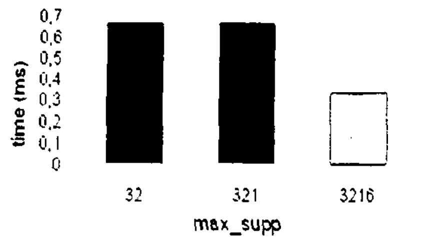
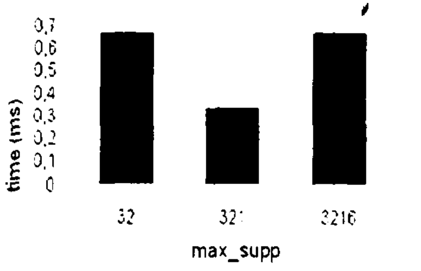
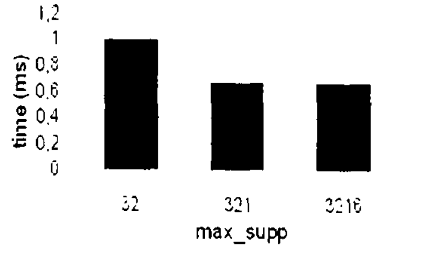
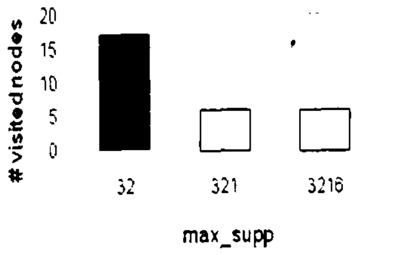
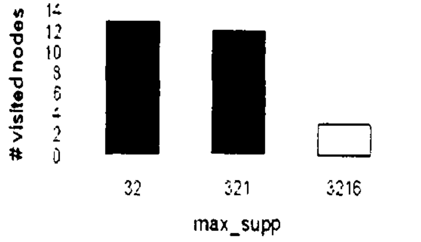
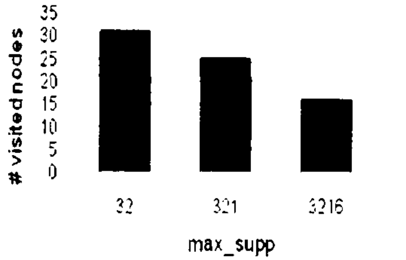
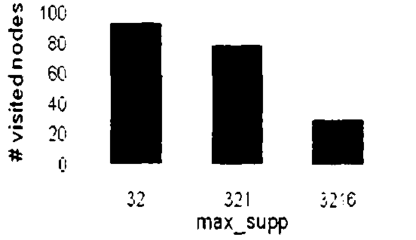
In this subsection we discuss the performance of the algorithm PartialLatticeNegotiation in terms of time and visited nodes. Our findings are as follows:

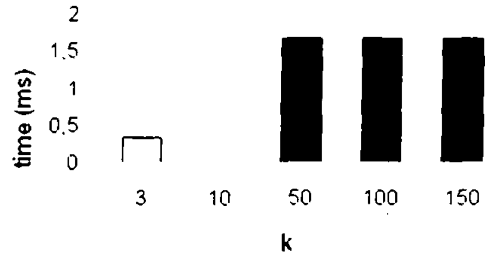
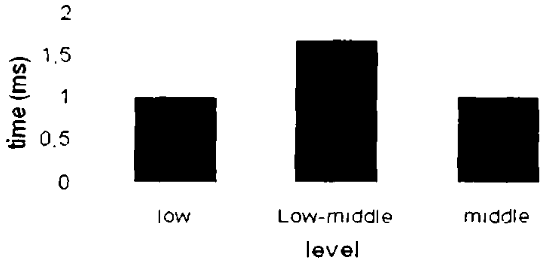
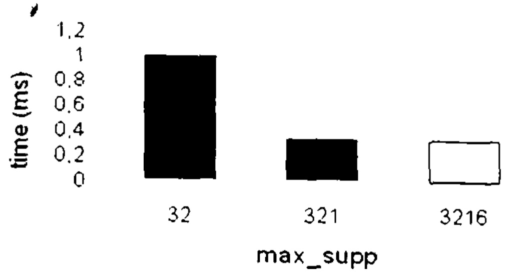
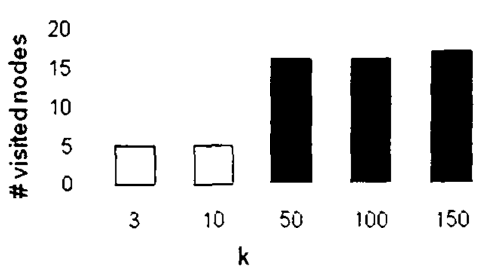
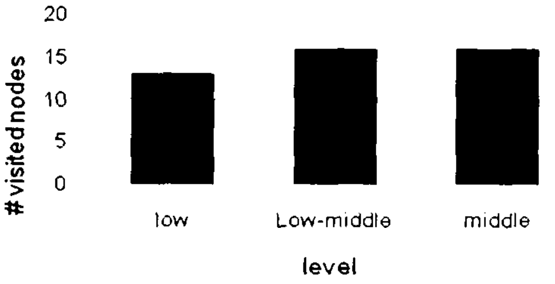
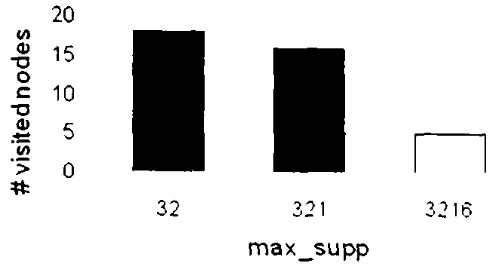
- The full lattice was already a fast answering mechanism for the QI sizes that we have explored. In all occasions, the time to navigate over the full lattice before returning an answer was between 0.66 – 8 msec. In the case of the partial lattice, the times range between 0.33 to 2 msecs, due to the reduced "lattice" size. In cases where time is really critical then this scaling down with a scale factor between 2 and 8 can be useful.
- Exactly as in the case of the full lattice, the increase of k results in an increase in the number of visited nodes for the case where we resort to relaxations. Concerning the case of exact answers, although our experiment does not give conclusive answers on the behavior of the algorithm, it is noteworthy that out of the 3 cases in QI=3 where the full lattice gave an exact solution, the two were retained in the partial lattice too.
- The behavior of the algorithm over the full lattice as the height of the allowed exact solution rises is retained. As the height constraints are put higher, there are more nodes to be visited for exact solutions from this height downwards. On the contrary, when we have to resort to relaxations things remain quite stable. Here, it is noteworthy to discuss the role of the top acceptable node v_{max}. If v_{max} is present we can have a very quick test on whether we will need relations (in most case we will), or an exact answer is possible (this happens in the case of QI=4, height=low, for example). In general, however, this luxury is not always available in the partial lattice, and this increases the search space.
- The behavior of the algorithm over the full lattice as *maxSupp* increases is also retained: time drops as maxSupp increases, since we find a desired solution faster.
- In all cases, the dominant factor for the performance of the algorithm is again QI size; naturally the effect is scaled down as the lattice size is scaled down too. Interestingly, it is worth noting that the maximum number of nodes visited

by the algorithm over the partial lattice is 94 which is the 5.2% of the maximum number of visited nodes in the case of full lattice (1792).

<table>
<tr><th colspan="4">Variant k</th></tr>
<tr><th>|QI|=3</th><th>|QI|=4</th><th>|QI|=5</th><th>|QI|=6</th></tr>
<tr>
<td>
</td>
<td>
</td>
<td>
</td>
<td>
</td>
</tr>
<tr>
<td>
</td>
<td>
</td>
<td>
</td>
<td>
</td>
</tr>
<tr>
<td><b>Parameters:</b>3, 321, 2 1 1 <b>Solution:</b>
Move down find, id:4 supp tuple:125 lvl: 1 0 0
<b>Parameters:</b>10, 321, 2 1 1 <b>Solution:</b>
Move down find, id:5 supp tuple:170 lvl: 1 1 0
<b>Parameters:</b>50, 321, 2 1 1 <b>Solution:</b>
<b>Rlx1</b>: id:20 supp:673 lvl: 1 1 1
<b>Rlx2</b>: id:35 supp tpls:137 lvl: 1 1 2
<b>Rlx3</b>: id:20 k:31 supp:314 lvl: 1 1 1</td>
<td><b>Parameters:</b>3, 321, 2 0 1 1 <b>Solution:</b>
<b>Rlx1</b>: id:55 supp:635 lvl: 1 0 0 1
<b>Rlx2</b>: id:103 supp tpls:50 lvl: 1 1 0 2
<b>Rlx3</b>: No solution
<b>Parameters:</b>10, 321, 2 0 1 1 <b>Solution:</b>
<b>Rlx1</b>: id:55 supp:2349 lvl: 1 0 0 1
<b>Rlx2</b>: id:61 supp tpls:257 lvl: 1 2 0 1
<b>Rlx3</b>: No solution
<b>Parameters:</b>50, 321, 2 0 1 1 <b>Solution:</b>
<b>Rlx1</b>: id:55 supp:7669 lvl: 1 0 0 1
<b>Rlx2</b>: id:107 supp tpls:137 lvl: 1 2 1 2
<b>Rlx3</b>: No solution</td>
<td><b>Parameters:</b>3, 321, 2 1 0 1 2 <b>Solution:</b>
<b>Rlx1</b>: id:271 supp:3639 lvl: 1 0 0 0 1
<b>Rlx2</b>: id:187 supp tpls:126 lvl: 4 0 2 0 0
<b>Rlx3</b>: No solution
<b>Parameters:</b>10, 321, 2 1 0 1 2 <b>Solution:</b>
<b>Rlx1</b>: id:271 supp:9223 lvl: 1 0 0 0 1
<b>Rlx2</b>: id:188 supp tpls:274 lvl: 4 0 2 1 0
<b>Rlx3</b>: No solution
<b>Parameters:</b>50, 321, 2 1 0 1 2 <b>Solution:</b>
<b>Rlx1</b>: id:271 supp:19324 lvl: 1 0 0 0 1
<b>Rlx2</b>: id:636 supp tpls:169 lvl: 4 0 1 2 2
<b>Rlx3</b>: No solution</td>
<td><b>Parameters:</b>3, 321, 2 1 1 0 1 2 <b>Solution:</b>
<b>Rlx1</b>: id:321 supp:7226 lvl: 1 0 0 0 0 1
<b>Rlx2</b>: id:763 supp tpls:155 lvl: 4 0 0 2 0 2
<b>Rlx3</b>: No solution
<b>Parameters:</b>10, 321, 2 1 1 0 1 2 <b>Solution:</b>
<b>Rlx1</b>: id:321 supp:14627 lvl: 1 0 0 0 0 1
<b>Rlx2</b>: id:2805 supp tpls:300 lvl: 4 0 0 2 1 2
<b>Rlx3</b>: No solution
<b>Parameters:</b>50, 321, 2 1 1 0 1 2 <b>Solution:</b>
<b>Rlx1</b>: id:321 supp:24558 lvl: 1 0 0 0 0 1
<b>Rlx2</b>: id:2812 supp tpls:137 lvl: 4 0 1 2 2 2
<b>Rlx3</b>: No solution</td>
</tr>
</table>

<table>
<tr><th colspan="4">Variant level</th></tr>
<tr><th>|QI|=3</th><th>|QI|=4</th><th>|QI|=5</th><th>|QI|=6</th></tr>
<tr>
<td> </td>
<td> </td>
<td> </td>
<td> </td>
</tr>
<tr>
<td> </td>
<td> </td>
<td> </td>
<td> </td>
</tr>
<tr>
<td><b>Parameters</b>:10, 321, 1 0 1 <b>Solution:</b>
Move down find id19 supp tpl:257 lvl: 1 0 1
<b>Parameters</b>:10, 321, 2 1 1 <b>Solution:</b>
Move down find, id:5 supp tpl:170 lvl: 1 1 0
<b>Parameters</b>:10, 321, 2 1 2 <b>Solution:</b>
Move down find, id:5 supp tpl:170 lvl: 1 1 0</td>
<td><b>Parameters</b>:10, 321, 1 0 0 1 <b>Solution:</b>
<b>Rlx1</b>: id:55 supp:2349 lvl: 1 0 0 1
<b>Rlx2</b>: id:61 supp tpls:257 lvl: 1 2 0 1
<b>Rlx3</b>: No solution
<b>Parameters</b>:10, 321, 2 0 1 1 <b>Solution:</b>
<b>Rlx1</b>: id:55 supp:2349 lvl: 1 0 0 1
<b>Rlx2</b>: id:61 supp tpls:257 lvl: 1 2 0 1
<b>Rlx3</b>: No solution
<b>Parameters</b>:10, 321, 2 1 1 2 <b>Solution:</b>
Move down find, id:103 supp tuple:285 lvl: 1 1 0 2</td>
<td><b>Parameters</b>:10, 321, 1 1 0 0 1 <b>Solution:</b>
<b>Rlx1</b>: id:271 supp:9223 lvl: 1 0 0 0 1
<b>Rlx2</b>: id:188 supp tpls:274 lvl: 4 0 2 1 0
<b>Rlx3</b>: No solution
<b>Parameters</b>:10, 321, 2 1 0 1 2 <b>Solution:</b>
<b>Rlx1</b>: id:271 supp:9223 lvl: 1 0 0 0 1
<b>Rlx2</b>: id:188 supp tpls:274 lvl: 4 0 2 1 0
<b>Rlx3</b>: No solution
<b>Parameters</b>:10, 321, 2 2 1 1 2 <b>Solution:</b>
<b>Rlx1</b>: id:508 supp:2250 lvl: 1 1 1 0 2
<b>Rlx2</b>: id:635 supp tpls:60 lvl: 4 0 1 1 2
<b>Rlx3</b>: No solution</td>
<td><b>Parameters</b>:10, 321, 1 1 1 0 0 1 <b>Solution:</b>
<b>Rlx1</b>: id:321 supp:14627 lvl: 1 0 0 0 0 1
<b>Rlx2</b>: id:2805 supp tpls:300 lvl: 4 0 0 2 1 2
<b>Rlx3</b>: No solution
<b>Parameters</b>:10, 321, 2 1 1 0 1 2 <b>Solution:</b>
<b>Rlx1</b>: id:321 supp:14627 lvl: 1 0 0 0 0 1
<b>Rlx2</b>: id:2805 supp tpls:300 lvl: 4 0 0 2 1 2
<b>Rlx3</b>: No solution
<b>Parameters</b>:10, 321, 2 2 2 1 1 2 <b>Solution:</b>
<b>Rlx1</b>: id:588 supp:3971 lvl: 1 0 2 1 0 2
<b>Rlx2</b>: id:2811 supp tpls:54 lvl: 4 0 1 2 1 2
<b>Rlx3</b>: No solution</td>
</tr>
</table>

Variant max supp			
|QI|=3	|QI|=4	|QI|=5	|QI|=6
			
			
Parameters:10, 32, 2 1 1 **Solution:** **Rlx1**: id:20 supp:77 lvl: 1 1 1 **Rlx2**: id:35 supp tpls:29 lvl: 1 1 2 **Rlx3**: id:20 k:5 supp:32 lvl: 1 1 1 **Parameters**:10, 321, 2 1 1 **Solution:** Move down find, id:5 supp tpl:170 lvl: 1 1 0 **Parameters**:10, 3216, 2 1 1 **Solution:** Move down find, id:1 supptpl:1921 lvl: 0 0 0	**Parameters**:10, 32, 2 0 1 1 **Solution:** **Rlx1**: id:55 supp:2349 lvl: 1 0 0 1 **Rlx2**: id:107 supp tpls:29 lvl: 1 2 1 2 **Rlx3**: No solution **Parameters**:10, 321, 2 0 1 1 **Solution:** **Rlx1**: id:55 supp:2349 lvl: 1 0 0 1 **Rlx2**: id:61 supp tpls:257 lvl: 1 2 0 1 **Rlx3**: No solution **Parameters**:10, 3216, 2 0 1 1 **Solution:** Move down find, id:55 supp tuple:2349 lvl: 1 0 0 1	**Parameters**:10, 32, 2 1 0 1 2 **Solution:** **Rlx1**: id:271 supp:9223 lvl: 1 0 0 0 1 **Rlx2**: id:638 supp tpls:14 lvl: 4 0 2 1 2 **Rlx3**: No solution **Parameters**:10, 321, 2 1 0 1 2 **Solution:** **Rlx1**: id:271 supp:9223 lvl: 1 0 0 0 1 **Rlx2**: id:188 supp tpls:274 lvl: 4 0 2 1 0 **Rlx3**: No solution **Parameters**:10, 3216, 2 1 0 1 2 **Solution:** **Rlx1**: id:271 supp:9223 lvl: 1 0 0 0 1 **Rlx2**: id:187 supp tpls:578 lvl: 4 0 2 0 0 **Rlx3**: id:271 k:2 supp:2211 lvl: 1 0 0 0 1	**Parameters**:10, 32, 2 1 1 0 1 2 **Solution:** **Rlx1**: id:321 supp:14627 lvl: 1 0 0 0 0 1 **Rlx2**: id:2812 supp tpls:21 lvl: 4 0 1 2 2 2 **Rlx3**: No solution **Parameters**:10, 321, 2 1 1 0 1 2 **Solution:** **Rlx1**: id:321 supp:14627 lvl: 1 0 0 0 0 1 **Rlx2**: id:2805 supp tpls:300 lvl: 4 0 0 2 1 2 **Rlx3**: No solution **Parameters**:10, 3216, 2 1 1 0 1 2 **Solution:** **Rlx1**: id:321 supp:14627 lvl: 1 0 0 0 0 1 **Rlx2**: id:1525 supp tpls:1222 lvl: 4 0 0 2 1 0 **Rlx3**: No solution

<table>
<tr><th colspan="3">IPUMS k-anonymity (QI=4)</th></tr>
<tr><th>Variant k</th><th>Variant level</th><th>Variant max supp</th></tr>
<tr>
<td>
</td>
<td>
</td>
<td>
</td>
</tr>
<tr>
<td>
</td>
<td>
</td>
<td>
</td>
</tr>
<tr>
<td><b>Parameters:</b>3, 6000, 2 1 1 0 <b>Solution</b>
Move down find, id:6 supp tuple:4462 lvl: 0 1 0 0
<b>Parameters:</b>10, 6000, 2 1 1 0 <b>Solution</b>
Move down find, id:26 supp tuple:3778 lvl: 1 1 0 0
<b>Parameters:</b>50, 6000, 2 1 1 0 <b>Solution</b>
<b>Rlx1</b>: id:27 supp:19599 lvl: 1 1 1 0
<b>Rlx2</b>: id:36 supp_tuples:2037 level: 1 3 0 0
<b>Rlx3</b>: id:27 k:17 supp:5748 lvl: 1 1 1 0
<b>Parameters:</b>100, 6000, 2 1 1 0 <b>Solution</b>
<b>Rlx1</b>: id:27 supp:41895 lvl: 1 1 1 0
<b>Rlx2</b>: id:36 supp_tuples:4775 level: 1 3 0 0
<b>Rlx3</b>: id:27 k:17 supp:5748 lvl: 1 1 1 0
<b>Parameters:</b>150, 6000, 2 1 1 0 <b>Solution</b>
<b>Rlx1</b>: id:27 supp:64389 lvl: 1 1 1 0
<b>Rlx2</b>: id:86 supp_tuples:2482 level: 4 1 0 0
<b>Rlx3</b>: id:27 k:17 supp:5748 lvl: 1 1 1 0</td>
<td><b>Parameters:</b>50, 6000, 1 0 1 0 <b>Solution</b>
<b>Rlx1</b>: id:21 supp:142078 lvl: 1 0 0 0
<b>Rlx2</b>: id:36 supp_tuples:2037 level: 1 3 0 0
<b>Rlx3</b>: id:21 k:3 supp:5688 lvl: 1 0 0 0
<b>Parameters:</b>50, 6000, 2 1 1 0 <b>Solution</b>
<b>Rlx1</b>: id:27 supp:19599 lvl: 1 1 1 0
<b>Rlx2</b>: id:36 supp_tuples:2037 level: 1 3 0 0
<b>Rlx3</b>: id:27 k:17 supp:5748 lvl: 1 1 1 0
<b>Parameters:</b>50, 6000, 2 2 2 0 <b>Solution</b>
<b>Rlx1</b>: id:27 supp:19599 lvl: 1 1 1 0
<b>Rlx2</b>: id:186 supp_tuples:207 level: 4 1 0 1
<b>Rlx3</b>: id:27 k:17 supp:5748 lvl: 1 1 1 0</td>
<td><b>Parameters:</b>50, 600, 2 1 1 0 <b>Solution</b>
<b>Rlx1</b>: id:27 supp:19599 lvl: 1 1 1 0
<b>Rlx2</b>: id:86 supp_tuples:527 level: 4 1 0 0
<b>Rlx3</b>: id:27 k:3 supp:476 lvl: 1 1 1 0
<b>Parameters:</b>50, 6000, 2 1 1 0 <b>Solution</b>
<b>Rlx1</b>: id:27 supp:19599 lvl: 1 1 1 0
<b>Rlx2</b>: id:36 supp_tuples:2037 level: 1 3 0 0
<b>Rlx3</b>: id:27 k:17 supp:5748 lvl: 1 1 1 0
<b>Parameters:</b>50, 60000, 2 1 1 0 <b>Solution</b>
Move down find, id:26 supp tuple:30578 lvl: 1 1 0 0</td>
</tr>
</table>

5.7. The effect of the number of selected nodes

Insofar, we have explored the case where we restrict the number of selected nodes to 5% of the full lattice (to be exact, to at least two nodes per height and 5% of the height's nodes otherwise). However, we have not explored the case where we modify this selectivity parameter to other values. To this end, we have explored to other values with a reasonable amount of selectivity, specifically 1% and 10%. We believe that given the antagonizing goals of fast lattice pre-computation and reasonably constraint deviation from the optimal solutions, these seem quite appropriate limits for the selectivity factor.

We varied *p*% to the values 1%, 5%, 10% and *k* to the typically used values *k*=3,10,50 and observed the results. The results are astonishing:

- The exact answers and the relaxation of *maxSupp* (approximation 1) or *k* (approximation 3) are identical in all three values of *p*%, for all values of *k*.
- The relaxation of height (Approximation 2), which explores all the available lattice, was slightly better when *p*% was raised to 10% and slightly worse when *p*%=1%. In Fig. 5.22, we depict the differences of 1% and 10% with respect to 10% with dark background and white font.

Based on the above, we can argue that a reasonable value for the selectivity factor between 1% and 10% suffices to provide the same results without further tuning.

p=1%	k=3	k=10	k=50		Approx 2	k=3	k=10	k=50		Approx 2	k=3	k=10	k=50
QI=3			-114		QI=3	-	-	0		QI=3	-	-	0
QI=4	-233	87	0		QI=4	1	0	0		QI=4	0	0	0
QI=5	18	-89	81		QI=5	0	1	1		QI=5	0	0	0
QI=6	0	15	-58		QI=6	0	0	1		QI=6	0	0	0
p=5%	k=3	k=10	k=50		Approx 2	k=3	k=10	k=50		Approx 2	k=3	k=10	k=50
QI=3			-114		QI=3	-	-	0		QI=3	-	-	0
QI=4	-233	87	0		QI=4	1	0	0		QI=4	0	0	0
QI=5	18	44	0		QI=5	0	0	0		QI=5	0	0	0
QI=6	0	15	0		QI=6	0	0	0		QI=6	0	0	0
p=10%	k=3	k=10	k=50		Approx 2	k=3	k=10	k=50		Approx 2	k=3	k=10	k=50
QI=3			-114		QI=3	-	-	0		QI=3	-	-	0
QI=4	-255	0	0		QI=4	0	0	0		QI=4	0	0	0
QI=5	0	0	0		QI=5	0	0	0		QI=5	0	0	0
QI=6	0	0	0		QI=6	0	0	0		QI=6	0	0	0

Figure 5.22 Differences for height relaxation (Approximation 2) for different values of p%

5.8. Extending the partial lattice at runtime

A critical factor that differentiates the full-lattice and the partial lattice methods is the existence of the histogram of the top-acceptable node. As we have seen, the partial lattice methods approximates the full lattice method quite well when (a) an exact answer can be found and (b) when we relax the height constraint and the search is expanded throughout all the available lattice. On the other hand, the partial-lattice method suffers at the relaxations of *k* and *MaxSupp*, which are exactly the ones that are executed over the top-acceptable node and nowhere else.

Therefore, it is clear that the presence of the histogram of the top-acceptable node would ameliorate the quality of the provided relaxations. Of course, this comes at the price of constructing the node's histogram at run-time. How severely is performance degraded if we pay the price of runtime construction to gain the high quality of solutions?

We have experimented with this extension. The algorithm Partial Lattice Negotiation is altered by adding the computation of the histogram of the top-acceptable node as the first step of the algorithm and restricting the approximations 1 and 3 to this node.

```
Algorithm PartialLatticeWithTopAcceptableHisto(L,k,h,MaxSupp)
In: Partial lattice L with the histograms for R,H, constraints for k, h, MaxSupp
Out: an exact solution s[v,k,h,supp] or s1,s2,s3, si=[v_i,k_i,h_i,supp_i]
Begin
Compute the histogram of v_max if not already in L;
if v_max  gives exact answer{
        for every height h, in height(v_max)-1, down to 0{
            for every node v in h, v in descendants(v_max), {
                if an exact answer is given by v
                    keep the v with the minimum suppression as v_opt;
                    (break ties by h)
            }
        }
else{
        approxSol_1 = ApproximateMaxSupp(L,v_max,k,h,MaxSupp);
        approxSol_2=ApproximateH(L,v_max,height(v_max),height(top),k,h,MaxSupp);
        approxSol_3 = ApproximateK(L,v_max,k,h,MaxSupp);

        return approxSol_1, approxSol_2, approxSol_3;
}
End.
```

Figure 5.23 Algorithm Partial Lattice With Top Acceptable Histogram

We have experimented by keeping *p*% to 5%. The results of our experimentation are very interesting. In a nutshell, the introduction of the computation of the histogram for the top-acceptable node introduces a significant overhead compared to the simple case of the partial lattice of the order of 140 – 335 msec, but, all the solutions practically coincide with the ones of the full lattice. Specifically, our results are as follows:

- The cases where an exact answer was given by the full lattice are all captured (as opposed to the three misses of the simple partial lattice). Out of these 12 occasions, there are two occasions where there is a discrepancy between the answer of the full lattice and the answer of the extended partial lattice.
- The relaxation of height remains practically the same as with the case of the partial lattice; remember that this is the case where all the available lattice is explored for the lowest possible height where a solution exists.
- The relaxation of *maxSupp* provides significant improvements compared to the case of the simple lattice. As expected, all the deviations in terms of suppressed tuples disappear (remember that the relaxations of *maxSupp* and *k* are performed at the top-acceptable node).
- Similarly, the deviations in terms of suppression and *k* for the relaxation of *k* also disappear. Most importantly, all the cases where the partial lattice failed to follow the behavior of the full lattice have disappeared. Again, this is due to the fact that the relaxation of *k* takes place on the top-acceptable node too.
- In terms of time, it is clear that the time is practically stable and dominated by the cost of the computation of the histogram for the top-acceptable node. In Figures 5.24-5.26 we depict the time and the number of visited nodes for different size of QI and different *k*, level of topmost, and *MaxSupp*.

Overall, one can argue with safety that if the time to compute the histogram for the top-acceptable node can be tolerated at runtime (and for the case of our experiments we believe it does), then the quality of solution improves drastically.

	Time (msec)			
k	QI=3	QI=4	QI=5	QI=6
3	144,00	206,67	260,00	334,33
10	143,00	205,33	259,67	334,33
50	142,67	204,00	260,00	334,33
	# visited nodes			
3	7	4	10	47
10	7	6	19	74
50	7	7	25	88

Figure 5.24 Time and visited nodes for all QI and Variant *k*

	Time (msec)			
level	QI=3	QI=4	QI=5	QI=6
Low	153,33	197	259,33	327,66
Low-middle	145,33	202,66	267,33	325,33
middle	144	206,66	262	310,66
	# of visited nodes			
Low	4	8	23	74
Low-middle	7	6	19	74
middle	9	9	11	27

Figure 5.25 Time and visited nodes for all QI and Variant level

	Time(msec)			
Max_supp	QI=3	QI=4	QI=5	QI=6
32	142,33	193,33	270	333,66
321	139,66	193	269	333,33
3216	137,66	193	269	334
	# visited nodes			
32	5	7	26	88
321	7	6	19	74
3216	7	4	4	25

Figure 5.26 Time and visited nodes for all QI and Variant

D suppressed tuples					D height					D KL			
Exact	k=3	k=10	k=50		Exact	k=3	k=10	k=50		Exact	k=3	k=10	k=50
QI=3	0	0	0		QI=3	0	0	0		QI=3	0	0	0
QI=4	-142	-	-		QI=4	1	-	-		QI=4	0	-	-
QI=5	-	-	-		QI=5	-	-	-		QI=5	-	-	-
QI=6	-	-	-		QI=6	-	-	-		QI=6	-	-	-
Approx 1	k=3	k=10	k=50		Approx 1	k=3	k=10	k=50		Approx 1	k=3	k=10	k=50
QI=3	-	-	-		QI=3	-	-	-		QI=3	-	-	-
QI=4	-	0	0		QI=4	-	0	0		QI=4	-	0	0
QI=5	0	0	0		QI=5	0	0	0		QI=5	0	0	0
QI=6	0	0	0		QI=6	0	0	0		QI=6	0	0	0
Approx 2	k=3	k=10	k=50		Approx 2	k=3	k=10	k=50		Approx 2	k=3	k=10	k=50
QI=3	-	-	-		QI=3	-	-	-		QI=3	-	-	-
QI=4	-	87	0		QI=4	-	0	0		QI=4	-	0	0
QI=5	18	44	0		QI=5	0	0	0		QI=5	0	0	0
QI=6	0	15	0		QI=6	0	0	0		QI=6	0	0	0
Approx 3	k=3	k=10	k=50		Approx 3	k=3	k=10	k=50		Approx 3	k=3	k=10	k=50
QI=3	-	-	-		QI=3	-	-	-		QI=3	-	-	-
QI=4	-	0	0		QI=4	-	0	0		QI=4	-	0	0
QI=5	-	-	-		QI=5	-	-	-		QI=5	-	-	-
QI=6	-	-	-		QI=6	-	-	-		QI=6	-	-	-

Figure 5.27 Summary of Qos deterioration for variant k (with vmax histogram construction)

D suppressed tuples				D height				DK			
Exact	low	low-middle	middle	Exact	low	low-middle	middle	Exact	low	low-middle	middle
QI=3	0	0	0	QI=3	0	0	0	QI=3	0	0	0
QI=4	-	-	0	QI=4	-	-	0	QI=4	-	-	0
QI=5	-	-	-	QI=5	-	-	-	QI=5	-	-	-
QI=6	-	-	-	QI=6	-	-	-	QI=6	-	-	-
Approx 1	low	low-middle	middle	Approx 1	low	low-middle	middle	Approx 1	low	low-middle	middle
QI=3	-	-	-	QI=3	-	-	-	QI=3	-	-	-
QI=4	0	0	-	QI=4	0	0	-	QI=4	0	0	-
QI=5	0	0	0	QI=5	0	0	0	QI=5	0	0	0
QI=6	0	0	0	QI=6	0	0	0	QI=6	0	0	0
Approx 2	low	low-middle	middle	Approx 2	low	low-middle	middle	Approx 2	low	low-middle	middle
QI=3	-	-	-	QI=3	-	-	-	QI=3	-	-	-
QI=4	87	87	-	QI=4	0	0	-	QI=4	0	0	-
QI=5	44	44	0	QI=5	0	0	0	QI=5	0	0	0
QI=6	15	15	0	QI=6	0	0	0	QI=6	0	0	0
Approx 3	low	low-middle	middle	Approx 3	low	low-middle	middle	Approx 3	low	low-middle	middle
QI=3	-	-	-	QI=3	-	-	-	QI=3	-	-	-
QI=4	-	-	-	QI=4	-	-	-	QI=4	-	-	-
QI=5	-	-	0	QI=5	-	-	0	QI=5	-	-	0
QI=6	-	-	0	QI=6	-	-	0	QI=6	-	-	0

Figure 5.28 Summary of Qos deterioration for variant height (with vmax histogram construction)

D suppressed tuples					D height					D k			
Exact	32	321	3216		Exact	32	321	3216		Exact	32	321	3216
QI=3	-	0	0		QI=3	-	0	0		QI=3	-	0	0
QI=4	-	-	239		QI=4	-	-	0		QI=4	-	-	0
QI=5	-	-	0		QI=5	-	-	0		QI=5	-	-	0
QI=6	-	-	-		QI=6	-	-	-		QI=6	-	-	-
Approx 1	32	321	3216		Approx 1	32	321	3216		Approx 1	32	321	3216
QI=3	0	-	-		QI=3	0	-	-		QI=3	0	-	-
QI=4	0	0	-		QI=4	0	0	-		QI=4	0	0	-
QI=5	0	0	-		QI=5	0	0	-		QI=5	0	0	-
QI=6	0	0	0		QI=6	0	0	0		QI=6	0	0	0
Approx 2	32	321	3216		Approx 2	32	321	3216		Approx 2	32	321	3216
QI=3	1	-	-		QI=3	0	-	-		QI=3	0	-	-
QI=4	1	87	-		QI=4	0	0	-		QI=4	0	0	-
QI=5	0	44	-		QI=5	0	0	0		QI=5	0	0	-
QI=6	0	15	0		QI=6	0	0	0		QI=6	0	0	0
Approx 3	32	321	3216		Approx 3	32	321	3216		Approx 3	32	321	3216
QI=3	0	-	-		QI=3	0	-	-		QI=3	0	-	-
QI=4	-	0	-		QI=4	-	0	-		QI=4	-	0	-
QI=5	-	-	-		QI=5	-	-	-		QI=5	-	-	-
QI=6	-	-	0		QI=6	-	-	0		QI=6	-	-	0

Figure 5.29 Summary of Qos deterioration for variant maxSupp (with vmax histogram construction)

5.9. Summary of findings

We have experimented for the effectiveness and efficiency of the proposed method over two data sets, the Adult and the IPUMS data sets, with the same parameters as we have experimented in the full lattice approach.

In summary, we can state the following about the simple partial lattice construction:

- The exact answering and the answering for the relaxation of height (that explores all the available lattice) provide very good approximations to the optimal solutions provided by the exact lattice. Specifically,
  (a) Only 3 out of 10 exact answers are missed and compensated by relaxations
  (b) The height relaxation has very small, or zero deviations from the suggestions of the full-lattice method.
- The relaxation of suppression provides answers that are gravitated towards the lower parts of the sublattice of the top-acceptable node and, thus, result in high values of suppression as compared to the ones provided by the top-acceptable node in the full lattice approach.
- The relaxation of k was already having a hard time finding answers in the full-lattice approach. This becomes worse in the partial-lattice approach and few results are returned.

Concerning the rest of the problem's parameters, we can state the following:

- The time needed to answer a user request ranges between 0.33 – 2 msecs for the case of simple partial lattice
- The increase of *k* increases the search space for the relaxations; the same happens as the *maxSupp* is decreased
- The size of QI is a determining factor for the behavior of the proposed method. Observe that small QI sizes give exact answers. At the same time, the size of the partial lattice, and consequently, the time to construct its histograms is proportional to the selectivity factor. For example, in the case of QI=6 with *p* = 5%, the lattice size is 94 – i.e., the 5.2% of the full lattice with 1792 nodes.

The extension that computes the histogram of the top-selection node at runtime results in

  - a time penalty of 0.1 – 0.3 sec;
  - a drastic improvement of the two relaxations that suffered in the previous approach (identical behavior to the case of the full lattice);
- small improvement for the exact answers and no improvement for the relaxation of height.

CHAPTER 6. RELATED WORK

The problem of preserving data privacy has been extensively studied both in the past and in the recent literature. Previous to the '00s, the largest part of research was conducted in the context of statistical databases, and several techniques have been proposed that involve swapping values and adding noise to the data in order to meet a general statistical property [AdWo89]. During the '00s, the area received a renewed interest by the research community. In this section, we cover the most important papers that are related to our approach; we refer the interested reader to the excellent survey of Fung et al. [FWCY10] for further probing.

Privacy in the field of data management deals with the problem of concealing sensitive information about individual records without destroying the data mining utility of the published data set. Take for example the case of medical records of a relation *T(Name,Age,ZipCode,Disease)* that is to be exported to analysts for data mining purposes. On the one hand, our aim is to provide the analysts with as much statistically important information as possible; on the other hand, we want to hide the relationship of individuals (identified by the *identifier* attribute *Name*) with the *sensitive* attribute *Disease*. This equilibrium among goals is primarily achieved by removing the statistically insignificant attribute *Name* from the published version of the relation. Unfortunately, it is still possible to breach the individuals' privacy via *quasi-identifier* attributes (in our example, *Age* and *ZipCode*) which can convey contextual information to an attacker about the concealed identifier attributes and their linkage to sensitive attributes (in our example, a patient's neighbor who knows the zip code and age of a patient can reason on the patient's disease if there are no other patients with similar characteristics). For this purpose three main families of techniques has been presented to preserve data:

- Domain generalization which is the main technique explored by the research literature with several sub-categories.
- Perturbation and control introduction of noise
- Anatomization of the published relation to separate the coexistence of quasi-identifiers and sensitive values in the same published record

6.1. Alterative techniques

The last of these methods (also latest in terms of when they were introduced), is a*natomization*. Anatomization dictates that we should not seek to modify the quasi-identifiers or the sensitive attribute, but, rather, it de-associates the relationship between them. So, we organize records in groups, each group with a variant set of sensitive values and we publish two tables: one with the sensitive values of each group and another with the quasi-identifiers and a group id in the place of the sensitive value. In Figure 6.1 we demonstrate the effect of anatomizing the data of table in Figure 2.1. Unfortunately, the data presented by anatomization are not very helpful for the well-meaning users due to their nature (remember that the published data can have thousands of records).

Age	Work_class	Education	Group ID
39	Private	Hs-grad	1
38	Private	Hs-grad	1
37	Private	Hs-grad	1
38	Private	11th	1
28	Loc-gov	Bachelors	1
31	Federal-gov	Master	2
30	State-gov	Bachelors	2
32	Self-emp-not-inc	Bachelors	2
35	Self-emp-inc	Prof-school	2
33	Self-emp-inc	Assoc-acd	2

(a)

Group ID	Hours per week	Count
1	40	2
1	50	1
1	45	1
1	30	1
2	50	2
2	60	1
2	54	1
2	40	1

(b)

Figure 6.1 Anatomization: (a) quasi identifier table, and, (b) sensitive attribute.

A second method for the publishing of data involves the *perturbation* of tuples. These means that we distort the sensitive values of the published tuples while keeping the statistical properties of the published data set as close as possible to the ones of the original data set. We refer the interested reader to [FWCY10] for a detailed survey of the works in this area. The main problems with perturbation are that (a) the published data contain noise (sometimes statistically significant) and it is possible that the well-meaning data analysts are annoyed by its presence and (b) the noise introduction is performed in a way that retains a specific statistical property, thus resulting in sometimes significant deviations for any other statistical measure of the published data set.

6.2. Generalization

The third area, provides a privacy-preserving version of original data by replacing the values of the original table with abstractions (e.g., a value of 451** for zip code instead of 45110). The ultimate goal in terms of privacy is to conceal each individual tuple into an appropriately constructed group of data, in a way that an attacker cannot easily reason about the participation of individuals into the group.

This method is called *generalization* as it iteratively generalizes the values of the published data set in higher levels of abstraction until the desired level of privacy is attained. In every step of this process, each individual tuple becomes a member of a larger group of tuples that all share the same quasi-identifier values ('hidden in the crowd'). If the data set is almost capturing the privacy criterion for most of its groups and there are only few groups that violate it, instead of generalizing again, it is possible to resort in the removal of the tuples of these outlier groups. This process is called *suppression*. The area of generalization is organized in three sub-areas.

- Full-domain generalization, or global recoding
- Multidimensional recoding
- Local recoding

The three main classes of works to which the related literature around data generalization is classified, all have their own characteristics, along with advantages and disadvantages. Full domain generalization or global recoding assumes a fixed set of anonymization levels to which values are generalized. Each quasi identifier comes with its own hierarchy of anonymization levels and mappings of values. For example, ages can come in years, 5 year periods, 10 year periods; in accordance to this scheme at the schema level, age 23 at the year level is mapped to the interval [21,25] at the 5-year level and the interval [21,30] at the 10-year level. On the other hand, multidimensional and local recoding instead of trying to create groups according to these hierarchies, they work in the opposite direction: they exploit the distribution of data in the multidimensional space in order to create the groups.

Formally, assume a relation T that is to be published as a transformed relation T^*.. The *semantics* of the generalization process can be regarded as the execution of two steps:

(a) First, the employed method partitions T to a set of disjoint groups, $\mathbf{P} = \{P_1 \text{ U } P_2 \text{ U } \ldots P_m\}$, such that the privacy constraint holds for each group.

(b) Then, T^* is produced by removing the identifier attribute from T and replacing the values of the quasi identifier attributes with a characteristic representation; this is typically the generalized variants of the microdata values (e.g., replace zip code 45110 with 451**).

Note that this is the fundamental intuition of the process and not necessarily the algorithmic steps to be followed.

The different categories of the generalization family of algorithms are distinguished mainly by the way they partition data. Global recoding replaces values independently of their group, whereas local as well as multidimensional recoding replace values with respect to the contents of the group. The difference of multidimensional from local recoding is that the former groups tuples in disjoint regions of the multidimensional space, whereas local recoding allows dense regions to "lend" data to sparse regions so that every group satisfies the privacy constraint. The replacement is typically done either by using a predefined hierarchy or by taking the *minimum bounding box* of the region in the multidimensional space; however, other presentation methods can be devised too (such as the choice of representative values from each group). Observe that in terms of our formal definition, the constructed groups are not necessarily equivalence classes: in local recoding, two tuples with same quasi identifier values may end up in different groups and different replacement (i.e., anonymization) method.

Full-domain generalization is supported by quite efficient algorithms. The problem with full-domain generalization is that it generalizes sparse and dense regions of the multidimensional space in the same way. So, it generalizes all the data set to the generalization scheme needed by the weakest of its groups. To avoid this, suppression can be used, but then, the utility of the published data set diminishes as a (sometimes large) part of it is removed. On the contrary, multidimensional and local recoding avoid suppression and instead of aligning the groups of tuples to the level hierarchies, they align the bounds of the groups to the distribution of tuples in the multidimensional space. This is not so efficient as in the case of global recoding but provides higher utility for the detailed inspection of the tuples. Unfortunately, the data mining tools suffer since the data are not in a homogeneous level of abstraction and therefore the classification or association rules that are extracted miss information. At the same time, the users are not always happy with the grouping of tuples given by the local recoding algorithms, as they are accustomed to the semantically meaningful hierarchies that are used in the case of global recoding.

6.2.1. Full-domain generalization

Full domain generalization is quite fast, since the complexity of the anonymization process mainly depends on the combinations of levels, one per quasi identifier that must be tested. Here, we cover the (rather straightforward) case of k-anonymity quickly, and expand the case of l-diversity more.

K-anonymity. In [Sama01, Swee02] the fundamental notion of k-anonymity is introduced along with the techniques of generalization and suppression that are mainly used in order to transform the initial dataset to an anonymized one that meets the k-anonymity principle. From that time, there has been a large body of work that contributes to data privacy using several k-anonymization algorithms. In [BaAg05], the authors introduce an algorithm that provides an anonymization of the data set based on the total ordering of the domains of its attributes. The idea is that even categorical domains are mapped to integers and an iterative process examines all the possibilities of grouping these values in abstraction groups (via an enumeration tree). Every anonymization scheme is accompanied by the cost in terms of information loss;

due to monotonicity reasons, a node can be pruned if its descendants cannot meet the optimal cost. LeFevre et.al [LeDR05| have proposed a clear way to describe full-domain generalization and introduce Incognito, a sound and complete algorithm for producing k-anonymous full domain generalizations using bottom-up aggregation along generalization dimensions and a-priori computation [AgSr94]; we discuss Incognito in more detail later in this section since it has been the basis for the recursive construction of our lattices and their exploration.

L-diversity. The achievement of k-anonymity alone does not guarantee immunity to attacks: the authors of [MaGK06, MKGV07] present some severe privacy problems that can occur in a k-anonymized dataset when the distribution of values for the sensitive attribute within a group is small (a single value in the worst case); to alleviate the problem, the authors introduce l-diversity as a new privacy-aware principle. The main idea of the paper is to go beyond k-anonymity in ensuring that identifier attributes are not linked to their sensitive counterparts via background knowledge of the attacker. The two highlighted vulnerabilities of k-anonymity are (a) the possibility of a whole group to have the same sensitive value and (b) the possibility of having too few sensitive values in the same group. In both cases, the individuals are not 'hidden in the crowd' of their group since all (or, a large number of) the members of the group have the same sensitive value. If this is the case, if an attacker relates an individual with a certain group, then he can confer with high probability the sensitive values of the hidden individual.

L-diversity is a criterion that tells us whether a group is versatile enough in order to effectively hide its members by exploiting both a large number of members and a large number of 'well-represented' values. The purpose is that the probability of relating an individual with its sensitive values is low, even in the case where the attacker can identify the individual's group. The authors of [MaGK06] propose three ways to implement the term 'well-represented':

(i) the distinct number of sensitive values in a group should be higher than l

(ii) the entropy of each group should be higher than $log(l)$

(iii) recursive l-diversity is achieved for each group. Assume that we sort the values of an sensitive attribute by their frequency in the group; let r_1, r_2, ..., r_m be the respective frequencies. In this case, we require that the highest frequency (r_1) is not greater than the sum of the lowest [*l..m*] frequencies (r_l, ..., r_m), multiplied by a scale factor c. (In other words, the frequent values are not too frequent and the infrequent values are not too infrequent).

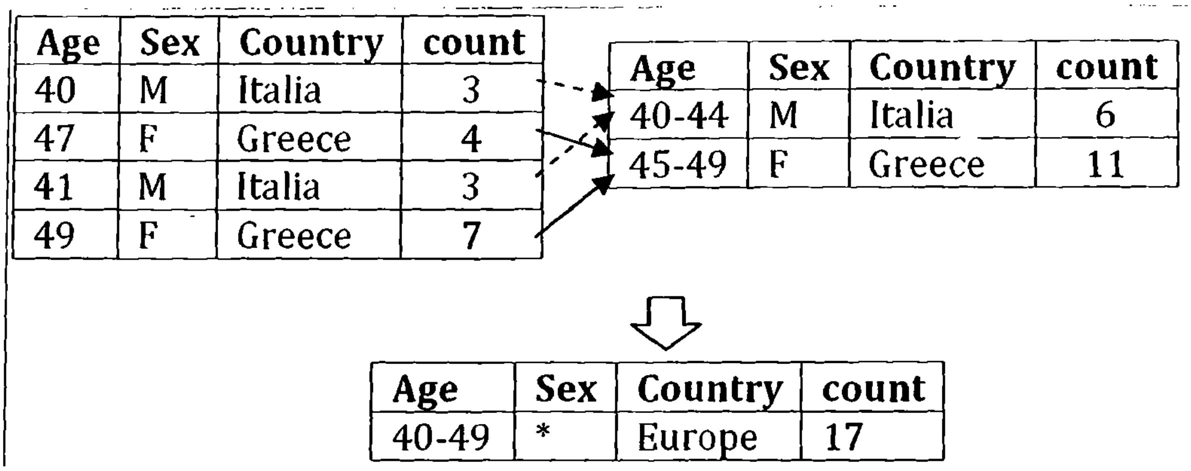
Incognito. The Incognito algorithm [LeDR05] is an efficient algorithm for the extraction of all the possible generalizations of a data set in order to achieve the criterion of k-anonymity or l-diversity.

Properties. The pillars of the algorithm are three important properties that characterize the nodes of the lattice and exploit the monotonicity of the hierarchies and the resulting groupings that derive from it. Specifically, *assuming a node of the lattice that is found to be k-anonymous*, these properties are:

- *Generalization*: Nodes found higher in the lattice that are derived from this node, are also k-anonymous
- *Rollup property*: Frequency sets of higher nodes can be computed from the current ones via the ancestor relationships of the involved values in their domain hierarchies
- *Subset property*: Nodes with fewer QI attributes are also anonymous

Specifically, the Generalization property dictates that if a relation $\underline{T}$ is k-anonymous over a set of quasi-identifier attributes $\underline{P}$, then $\underline{T}$ is also k-anonymous over a set of quasi-identifier attributes $\underline{Q}$ that are ancestors of the attributes of $\underline{P}$ in the respective hierarchies ($\underline{D_P \leq_D D_Q}$). In other words, if a node found low in the lattice qualifies for a solution, then, its ancestors also qualify as solutions. This is a simple outcome of the fact that the groups of the higher level node are produced by mergers of the groups at the lower level node; this results in fewer groups of larger cardinality.

The Rollup property states that once an ancestor node is a candidate solution, we can also compute its groups by exploiting the groups of any of the lower level nodes that are its descendants. Specifically, this is done by mapping the QI values of the descendant's groups to their respective values of the ancestor level; then, the frequency sets of all the descendant's groups that are mapped to the same ancestor group are summed to compute its frequency set. Observe Figure 6.2, where the values for the quasi-identifier *Age* (*A*), *Sex* (*S*), *Country* (*Z*) are rolled-up from the exact level of age (on the left of the figure) to the age level of 5-year intervals (on the right of the figure): the new frequency sets are simple sums of the respective frequency sets at the detailed level. The same applies again when we further anonymized the data set as depicted at the bottom of Figure 6.2.



Age	Sex	Country	count
40	M	Italia	3
47	F	Greece	4
41	M	Italia	3
49	F	Greece	7

Age	Sex	Country	count
40-44	M	Italia	6
45-49	F	Greece	11

Age	Sex	Country	count
40-49	*	Europe	17

Figure 6.2 Incognito's Rollup Property

Finally, the Subset property dictates that if you expand the quasi-identifier set with a new member (i.e., add an extra attribute to the QI set), then the groups are de-aggregated. The inverse is also useful, since the removal of an attribute from the QI, results in the merging of groups. Therefore, if a node is k-anonymous when the QI-set involves *N* attributes, then it is also k-anonymous with *N-1* of them as the quasi-identifier set. More importantly, in a manner that resembles Apriori pruning a lot, if a node is not k-anonymous when it is tested under *N* attributes as the quasi-identifier set, then there is no need to test it for k-anonymity for any superset of these *N* attributes, either.

Algorithm. Like all anonymization algorithms, Incognito uses as input the original data set (denoted as *T*), the set of attributes that constitute the quasi-identifier set (denoted as *QI*) along with their domain hierarchies and a value for the privacy criterion – here we use *k* for k-anonymity. The output of the algorithm is a graph that is a subset of the lattice formed by the Cartesian product of the domain hierarchies of the quasi-identifier set and contains all the generalizations that fulfill the input privacy criterion.

The crux of the algorithm is the stepwise expansion of the quasi-identifier set and the exploration of the respective intermediate lattices generated each time. The algorithm starts with all the possible QI-sets of size one (i.e., checks each attribute in isolation). Every intermediate lattice is visited via a breadth-first search, starting from the bottom all the way to the top. During this traversal, generalizations that fail to fulfill the k-anonymity criterion are pruned. This check is easily performed by counting the number of records per frequency set. Once all lattices of quasi-identifier size *N* have been explored, their subsets that survive the pruning process are combined to construct lattices of quasi-identifier size *N+1*. To generate these new lattices, the algorithm takes pairs of lattices of size *N* that are identical in the first *N-1* attributes and joins them. An interesting, Apriori-like, optimization is also the fact that for a node of QI-size *N* to be considered, *all* its generating nodes of QI-size *N-1* must have survived the process. This process terminates when the designated quasi-identifier set of attributes is explored.

Within this process, the aforementioned properties are exploited: if a lower-level node is found to be k-anonymous, all the nodes at higher levels of generalization that can be derived from it are marked as k-anonymous too. Moreover, the groups of higher-level nodes are produced by the groups of lower level nodes whenever this is possible.

The authors prove that the algorithm is sound (the solution generated is correct and does not violate the consistency constraints that a solution to the problem described is required to have) and complete (all correct solutions are returned).

Moreover, two extensions are also suggested:

(a) Due to the pruning process, some low level nodes are not part of the solution; however they can be reused to generate the rest of the surviving nodes. So, it is possible to pre-compute these 'super-root' nodes and avoid computing the lower parts of the output lattice all the way from the base relation.

(b) All possible subsets of the quasi-identifier size of the base-level generalization are pre-computed and re-used to avoid computing lower level solutions from the base relation.

6.2.2. Multidimensional and local recoding

Multidimensional recoding. Multidimensional recoding can be achieved via the Mondrian algorithm [LeDR06] which appears to be efficient and produces results with "more information at the browsing level" than global generalization. However, if one wishes to work with predetermined hierarchies, Mondrian is not suitable.

Local recoding. Fast algorithms for local recoding do exist [GhKM09]]; however, they do suffer from the same problem as Mondrian: they produce arbitrary (and, in fact, overlapping) regions for the grouping of source data. Algorithms performing local recoding with hierarchies are also available [Xu+06]; however, their performance is very slow for an on-line setting (Ghinita mentions that the Top-Down method of Xu et al takes around 2 hours for settings where the Hilbert Method of Ghinita et al., and the Mondrian method of Lefevre et al., take between 10 to 60 seconds; Xu et al in their KDD'06 paper mention: "...the runtime of the top-down approach is just less than 6 times slower than that of the MultiDim method.").

k-anonymity as spatial indexing. Iwuchukwu and Naughton [IwNa07] utilize an R-tree to speed up the anonymization process. The idea is that the internal nodes of the R-tree can be tuned in order to guarantee that the descendants of an internal node can always operate in groups of tuples of size no more than *k*. Once this is achieved, the anonymization process is very fast; in fact, it can also be easily tuned to the value of *k* the user desires for k-anonymity. The method operates well when an intuitive ordering of the detailed values can be achieved; in other words, whenever the domain of an attribute can be isomorphically mapped to the set of integers in an intuitive way, the R-tree approach is a very good solution. There are several benefits from the R-tree approach: it can be incrementally updated, it can be tuned to accommodate specific workloads fast, it can provide the aforementioned *multi-granular* anonymity and it provides good anonymizations very fast. At the same time, it is not straightforward how the method operates in categorical domains accompanied by hierarchies. In this case (which is also the case that we explore in our paper), it is not obvious that an internal node can always have a bounded number of descendants within the ranges required by the R-tree.

6.3. Suppression without generalization

Finally, despite the fact that the bulk of the research has been focused on various privacy criteria beyond k-anonymity and l-diversity, as well as towards efficient algorithms, mainly for local recoding, there exist some papers that explore the theoretical limits of optimal anonymization with respect to suppression and provide algorithms for its approximation. In this section we present these works as they seem to be the closest to our case.

Optimality and Approximate Algorithms for k-anonymity. The problem of finding the best possible anonymization scheme is in principle NP-hard. The theoretical foundations of the problem are given in [MeWi04]. The problem investigated in [MeWi04] is based on the idea of locally recoding a data set without reference to any hierarchies of values for the quasi-identifiers. Specifically, the problem is to try to minimize the number of cells (attention: cells, not tuples) that are suppressed (i.e., they take a value of *) in order to achieve k-anonymity. The authors of [MeWi04] prove that the problem is NP-hard and provide approximation bounds for it, based on the idea of the *diameter* of a set (which measures the maximum distance between any two tuples of the set, measured as the number of cells in which they differ). It is also interesting to note that the groups of the partition that are generated can be of bounded size: they are –of course- larger or equal than *k*, but they need not be larger than 2*k*-1. The authors provide an algorithm for the problem by adjusting a well-known greedy algorithm for the set cover problem to the setting of the problem. The set to be covered is the set of tuples of the table to be anonymized, say *T*, and therefore, the input to the algorithm is the set of all sets of tuples that are subsets of *T* whose cardinality is in the range of [*k*, 2*k*-1]. The greedy algorithm requires a penalty measure for each of these subsets that is selected each time and this is the diameter of the subset. The greedy algorithm results in a set cover of the original table *T*; since a cover is not a partition (i.e., a member of the original data set may be assigned to more than one of the covering sets) an adjustment must be made in order to turn the cover to a partition. The adjustment is simple test, applied repeatedly until no tuple belongs to two sets: if a tuple belongs to two sets, one of which is larger than *k*, then, it is removed from this set; if there are two sets to which the tuple belongs and they are both of size k, they are merged in one set.

Park and Shim in [PaSh07] extend the fundamental approximation approach of [MeWi04] with different levels of approximation. The authors still operate in the same setting as [MeWi04] — i.e., local recoding with no hierarchies and counting of suppressed cells— and start by changing the penalty for the greedy algorithm to operate on the basis of the *suppression length* of a set, which is the number of attributes where a value of * must be assigned, in any of the tuples of the set. Still, the previous approach suffers from the problem of having a too large input to generate. So, the authors of [PaSh07] extend this approach by observing that the frequent itemsets of the table T can serve as good starting point for identifying this input. The idea is that if a tuple t contains a frequent itemset that spans some of its attributes (which are called the representatives of t), it is possibly a good choice to leave them intact and consider the rest of the values as candidates for suppression. For each frequent itemset (frequent being the itemset with support larger than k in this paper) we compute the set of all tuples of T that contain it; this set is added to a set F_{FQ} which is inserted as input to the proposed algorithm. Several adjustments are also made in the algorithm, since it is possible that some of these sets are too large (larger than $2k$-1) than what is necessary. Moreover, the authors prove that instead of frequent itemsets, it is also possible to operate with closed frequent itemsets with the same approximation factor. In fact, the authors show that it is also possible to constrain the size of the suppression length by a factor of β with a bounded scale factor of β to the approximation factor. Finally, the authors provide a greedy algorithm that takes as input the subsets of T that are based on the closed frequent itemsets of T and sorts them with respect to their suppression length in increasing order. Then, the algorithm each time picks the next set of tuples and retains only its tuples not already covered; this new set is considered as a possible group of the final partition if its size is larger or equal than k. The authors have experimented with data sets of varying size; these experiments demonstrate that this last algorithm is significantly faster and provides a very good amount of suppression to the data set.

Curse of dimensionality on k-anonymity. In [Agg05] the author tries to prove that the amount of suppressed data increases more and more as the number of the attributes that can act as quasi-identifiers increases. The author starts with a theoretical analysis of how achieving k-anonymity via generalization relates to the probability that k-anonymity is violated for an arbitrary record in a data set. The author assumes that every attribute in the dataset can serve as a quasi-identifier. Also, the author assumes an identical normal distribution for a set of *d* independent dimensions of quantitative (i.e., not categorical) nature. Moreover, the method of generalization is reduced to replacing a tuple with the range of a surrounding "cell" around it (practically assigning a range of values for every dimension). Then, the author proves that the probability of achieving k-anonymity tends to zero as the number of dimensions rises to infinite. This practically means that since no data point in the data set can achieve k-anonymity at high dimensionalities, all the data set will have to be suppressed. Similarly, the second result of the paper relates to anonymization via clustering and demonstrates that as the number of dimensions tends to infinite, the replacement of a value by its appropriate cluster is meaningless, as the highest possible distance of two points in each cluster in the high-dimensional space is practically the same with the maximum distance of any two points in the whole data set. Finally, the author performs a simulation study for the aforementioned results and works with two data sets: (a) a synthetic data set containing 10000 points and 50 dimensions, generated in a way that the number of clusters can be regulated and (b) a market basket data set generated via the IBM generator, which contains data with higher skew. In both cases, the amount of suppressed tuples quickly rose from 0% in low dimensionalities to 80-90% in high dimensionalities for the simple case of 2-anonymity.

This is one the few papers dealing with the problem of suppression in k-anonymity. The paper is focused to the theoretical study of the effect of high dimensionality to the suppression; since these theoretical results demonstrate that high dimensionalities are rather prohibitive for anonymization, we have constrained ourselves to more practical settings that we have explored thoroughly. So, in our approach, we explore the problem taking into account various other parameters (hierarchies for the generalization, different values of k, different privacy criteria, and a more constrained approach to the dimensionality of the data sets, as compared to the theoretical limits of [Agg05]).

CHAPTER 7. CONCLUSIONS

The goal of this thesis has been to extend our documented knowledge and proposed on-line methods for the problem of privacy preserving data publishing. The ultimate goal pursuded by this thesis is to equip the data curator with the means to fine-tune several parameters around the privacy-preserving publishing of his data with other stakeholders by negotiating the values of suppression, generalization and privacy criterion in user-time, in order to quickly reach a consensus on the anonymization scheme among all interested stakeholders. Specifically, in this thesis we have attacked the following problems, not previously explored by the research community.

The first goal of this thesis has been to study the interplay of suppression, generalization and privacy criterion and record how changes to one of these parameters affect the two others. This would also determine whether the problem is worth investigating or not. We have worked with the criteria of k-anonymity and simple l-diversity over two data sets, the Adult and the IPUMS data set and our findings can be summarized as follows:

Overall, we can safely claim that the problem is valid and important. Low generalization heights (that are of more interest to us due to their information utility), or large values for the privacy criterion (which is of more interest to us due to the increased privacy it offers to individuals), or erroneous choice of generalization scheme can result in large amounts of suppressed data, quite possibly much higher as compared to more careful choices concerning the generalization scheme.

Our detailed findings concerning the relationship of the involved parameters can be summarized as follows:

- As the generalization height increases, the suppression drops quickly at small heights; the drop in suppression is less important in higher heights, where the number of suppressed tuples becomes statistically small and drops slowly. Interestingly, the overall trend for the decrease of suppression is practically the same for different values of *k* or *l* – of course, with different amounts of suppressed tuples.

- As the value for the privacy criterion (e.g., *k* in k-anonymity) increases, the suppression increases too. This is especially important in lower heights of generalization that are both important due to their information utility and demonstrate high volumes of suppression.
- As the size of the quasi identifier set increases, the effect to suppression is significant, as suppression increases too – sometimes drastically. Some quantitative evaluations around this theme suggest that (a) given a specific height and k an increase in QI size by one increases the suppression by a factor of 2 – 3; (b) to attain the same suppression threshold an increase in QI size by one, requires ascending 1-2 levels for k-anonymity and 2-3 levels for l-diversity.
- Not all attributes, generalization levels and, consequently, generalization schemes have the same effect to suppression. It is noteworthy that within the same height, the minimum possible suppression is approximately 2.5 times lower than the average for k-anonymity and 3 times lowers for l-diversity. This is especially evident in cases where the suppression has high values or values that cannot really be tolerated; on the other hand, for too large values of suppression (e.g., too large QIs or *k*) the relationship between average and minimum value does not follow this rule.
- Based on the above, it is important that for case that do matter, and where we can really attain good amounts for tuple suppression, it is really important to carefully pick the generalization scheme that will minimize this suppression. The faster we identify these generalization schemes the faster the process completes.

A second goal of this thesis was to provide efficient ways that allow the user achieve an anonymous data set with constraints over the generalization height, the amount of suppression and the tunable value of the privacy criterion. A third, related goal has *been the ability to provide suggestions to the user that are close to his original desideratum around generalization, suppression and privacy.*

We have attacked the above two goals via three methods. Both methods are based on precomputing statistical information for several possible generalization schemes (i.e., triplets of values for the minimum allowed value for the privacy criterion, the maximum allowed value for the generalization heights per attribute and the maximum tolerable amount of suppression). We organize generalization schemes in a lattice and compute histograms (appropriate to the employed privacy criterion) for the nodes of the lattice.

The first method we have employed pays the price to precompute the histograms for all the nodes in the lattice. Then, at runtime, the user gives as input three values, one for each of the abovementioned criteria as a desirable constraint. The algorithm we have introduced checks whether there exists a possible solution to the that satisfies all criteria and outputs either the scheme of lowest height that can respect all three criteria or, alternative schemes that provide relaxations to the user input. The three relaxations are based on the idea of keeping the two of the three values of the user input fixed and finding the closest possible approximation for the third parameter. We have proved that the proposed method is guaranteed to provide the best possible answers for the given user requests. Our experiments indicate that this is performed in less than 10 milliseconds for typical data sets used in the research literature.

However, the method comes at a price, and specifically, at the price of precomputing the histograms for all the nodes of the lattice. This precomputation requires several minutes (e.g., our experiments gave 20-40 minutes for the largest quasi-identifier sets). In one wishes to avoid this precomputation we provide a second method that precomputes only a small subset of the lattice's nodes with their histogram. To this end, we have also addressed the problem of which nodes of the lattice to select. Our approach is based on the ranking of generalization levels with respect to their grouping power (since, the larger the groups, the less the suppression). Then, we try to rank the combinations of levels for all the possible generalization schemes and pick a fixed subset of them (e.g., 5%). Our experiments demonstrate a linear speedup of the precomputation time with the approximation factor. The on-line answering has been sped up (due to the significantly smaller size of the lattice) and remain within few milliseconds per user request. At the same time, the quality of solution is quite good for (a) the case where an exact answer exists and (b) the relaxation requires exploring the full lattice. The price to pay however, is located in a couple of relaxations where the proposed solution is either gravitated towards lower nodes in the lattice (and provides, thus, solutions with high suppressions), or, fails to give an answer at all.

Finally, by observing that the two out of the three approximations are due to the top-acceptable node, we have proposed a third method that computes the histogram of this node at runtime. Based on our experiments, the time penalty for this extra computation is in the order of 0.1 – 0.3 sec and the two relaxations that suffered in the previous approach demonstrated an identical behavior to the case of the full lattice; therefore, if this time overhead can be tolerated in terms of user time (and for the case of our experiments we believe it does), then the quality of solution improves drastically.

Future work can take up on our results and explore alternative directions. A first possible way to go is the attempt to come up with some deeper understanding of the laws connecting the problem parameters and the measurable effects. So, experimentations over different data sets are required to observe the interrelationships of the parameters and how they affect the amount of suppression needed.
Second, we could extend the negotiation to other directions that could serve user needs. Maybe a user decided that some of the attributes make the negotiation difficult and wants to get rid of them. Or maybe, a user decides that the full domain generalization method that we support is not good for him and he would like to work with an alternative anonymization method. Maybe the user would like to have a quick preview on what results data mining tools can give for the anonymization scheme that our method proposes. All these possible user needs, provide unexplored turf for subsequent research

REFERENCES

[AdWo89] Adam N.T. and Wortman J.C. Security control methods for statistical databases. ACM Computer Surveys 21,4 (December) 1989.

[Agg05] Charu C.Aggarwal. On k-anonymity and the curse of Dimensionality. VLDB 2005.

[AgSr94] Agrawal, R. and R. Srikant. Fast Algorithms for Mining Association Rules, Proc. VLDB 1994

[AgST05] Rakesh Agrawal, Ramakrishnan Srikant, Dilys Thomas. Privacy Preserving OLAP. SIGMOD 2005

[BaAg05] Roberto J. Bayardo Jr., Rakesh Agrawal. Data Privacy through Optimal k-Anonymization. ICDE 2005

[FuWY05] Benjamin C. M. Fung, Ke Wang, Philip S. Yu: Top-Down Specialization for Information and Privacy Preservation. ICDE 2005 See also http://ddm.cs.sfu.ca

[FWCY10] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: a survey of recent developments. *ACM Computing Surveys (CSUR)* 2010

[GhKM09] Gabriel Ghinita, Panagiotis Karras, Panos Kalnis, Nikos Mamoulis. A framework for efficient data anonymization under privacy and accuracy constraints. ACM Trans. Database Syst. 2009

[IPUMS] Data set obtained from the web site of Y. Tao for the [XiTa07] paper http://www.cse.cuhk.edu.hk/~taoyf/paper/sigmod07.html

[IwNa07] Tochukwu Iwuchukwu, Toward Scalable and Incremental Anonymity. K-anonymity as Spatial Indexing: Toward Scalable and Incremental Anonymity. VLDB 2007

[LeDR05] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In SIGMOD 2005.

[LeDR06] Kristen LeFevre, David J. DeWitt, Raghu Ramakrishnan. Mondrian Multidimensional K-Anonymity. ICDE 2006

[LiLV07] Ninghui Li, Tiancheng Li, Suresh Venkatasubramanian t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. ICDE 2007

[LWFP08] Jiuyong Li, Raymond Chi-Wing Wong, Ada Wai-Chee Fu, Jian Pei. Anonymization by Local Recoding in Data with Attribute Hierarchical Taxonomies. IEEE Trans. Knowl. Data Eng 2008

[MaGK06] A. Machanavajjhala, J. Gehrke, and D. Kifer. l-diversity: Privacy beyond k-anonymity. ICDE 2006.

[MeWi04] A. Meyerson, R. Williams. On the complexity of optimal k-anonymity. PODS 2004.

[MKGV07] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, Muthuramakrishnan Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. ACM Transactions on Knowledge Discovery from Data. TKDD 2007

[PaSh07] Hyoungmin Park, Kyuseok Shim. Approximate Algorithms for k-anonymity. SIGMOD 2007.

[Sama01] P. Samarati. Protecting respondents' identities in microdata release. IEEE Trans. Knowl. Data Eng. TKDE 2001.

[Swee02a] Latanya Sweeney. k-Anonymity: A Model for Protecting Privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 2002

[Swee02b] Latanya Sweeney. Achieving k-Anonymity Privacy Protection Using Generalization and Suppression. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 2002

[UCI] U.C. Irvine Repository of Machine Learning Databases. 1998. http://www.ics.uci.edu/~mlearn

[XiTa06] X. Xiao and Y. Tao. Anatomy: Simple and effective privacy preservation. VLDB 2006.

[XiTa07] X. Xiao and Y. Tao. m-Invariance: Towards Privacy Preserving Re-publication of Dynamic Datasets. SIGMOD 2007.

[Xu+06] Jian Xu, Wei Wang, Jian Pei, Xiaoyuan Wang, Baile Shi, Ada Wai-

Chee Fu. Utility-based anonymization using local recoding. KDD 2006

[ZKSY07] Qing Zhang, Nick Koudas, Divesh Srivastava, Ting Yu. Aggregate Query Answering on Anonymized Tables. ICDE 2007

SHORT CV

Alexandra Pilalidou was born in 1984 and finished high school in 2003. She obtained his B.Sc. in Computer Science in 2008 from the computer Science Department of the University of Ioannina. Then she entered the Graduate Program of the same institution under the supervisor of Panos Vassiliadis. Her search research interests include the areas of database systems, with particular emphasis on privacy issues and software engineering.